

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



IMPLEMENTACIÓN DE MARCO DE SEGURIDAD DE
APLICACIONES EN NOT WASTE

FABIÁN ALEJANDRO BUSTOS AGUIRRE

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO EN INFORMÁTICA

Profesor Guía: José Luis Carrasco Sáez

Enero - 2025

DEDICATORIA

Dedico este proyecto a mi madre y a mi padre, en reconocimiento a su apoyo incondicional. También espero que este trabajo sea de utilidad a las personas del área de informática y ciberseguridad.

AGRADECIMIENTOS

Quiero expresar mi más sincero agradecimiento a mi mamá, a mi abuelo, mi padre, mi hermano, mi novia y toda mi familia, cuyo apoyo incondicional y constante aliento han sido fundamentales para la realización de este proyecto. Gracias por creer en mí y por estar siempre a mi lado.

RESUMEN

Resumen— Las startups suelen enfocarse en el desarrollo rápido de productos innovadores, dejando la ciberseguridad en un segundo plano. Sin embargo, asegurar la seguridad desde el inicio es fundamental para proteger los datos, generar confianza y evitar riesgos legales. La falta de recursos, conocimiento y experiencia en normativas hace que este desafío sea complejo, pero metodologías como OWASP SAMM proporcionan un enfoque estructurado y escalable. Este trabajo toma como caso de estudio a Not Waste, una startup que enfrenta dos problemas clave: el desperdicio de alimentos, su principal objetivo, y la amenaza de la ciberdelincuencia, común en empresas emergentes. Se propone adoptar un enfoque metodológico sólido en seguridad informática, con el fin de ofrecer una guía práctica para startups tecnológicas, ayudándolas a implementar una postura de seguridad estructurada que les permita crecer de manera sostenible y enfrentar con éxito los retos de seguridad actuales.

Palabras Clave—*Ciberseguridad; SAMM; Metodología; Desperdicio; Django.*

ABSTRACT

Abstract— Startups often focus on the rapid development of innovative products, putting cybersecurity on the back burner. However, ensuring security from the start is crucial to protect data, build trust, and avoid legal risks. The lack of resources, knowledge, and experience with regulations makes this challenge complex, but methodologies like OWASP SAMM provide a structured and scalable approach. This work uses Not Waste as a case study, a startup facing two key issues: food waste, its main goal, and the threat of cybercrime, common in emerging businesses. It proposes adopting a solid methodological approach to cybersecurity, aiming to provide a practical guide for tech startups, helping them implement a structured security posture that allows them to grow sustainably and successfully tackle current security challenges.

Keywords—*Cybersecurity; SAMM; Methodology; Waste; Django.*

GLOSARIO

- *API: Application Programming Interface.*
- *APP: Application.*
- *ASVS: Application Security Verification Standard.*
- *AWS: Amazon Web Service.*
- *BSIMM: Building Security In Maturity Model.*
- *CD: Continuous Deployment.*
- *CI: Continuous Integration.*
- *CMM: Capability Maturity Model.*
- *CSRF: Cross-Site Request Forgery.*
- *EC2: Elastic Compute Cloud.*
- *GCO: Global Cybersecurity Outlook.*
- *GDPR: General Data Protection Regulation.*
- *HTTP: Hypertext Transfer Protocol.*
- *IBM: International Business Machines Corporation.*
- *IAM: Identity and Access Management.*
- *IEC: International Electrotechnical Commission.*
- *IDS/IPS: Intrusion Detection System/Intrusion Prevention System.*
- *ISO: International Organization for Standardization.*
- *JSON: JavaScript Object Notation.*
- *JWT: JSON Web Token.*
- *MASVS: Mobile Application Security Verification Standard.*
- *NIST: National Institute of Standards and Technology.*

- *NIST CSF: National Institute of Standards and Technology Cybersecurity Framework.*
- *OWASP: Open Web Application Security Project.*
- *RDS: Relational Database Service.*
- *SAMM: Software Assurance Maturity Model.*
- *SAST: Static Application Security Testing.*
- *SDLC: Software Development Life Cycle.*
- *SSDLC: Secure Software Development Life Cycle.*
- *UTFSM: Universidad Técnica Federico Santa María.*
- *VCS: Version Control System.*
- *WAF: Web Application Firewall.*
- *WEF: World Economic Forum.*

ÍNDICE DE CONTENIDOS

RESUMEN.....	3
ABSTRACT	3
GLOSARIO	4
ÍNDICE DE CONTENIDOS	6
ÍNDICE DE FIGURAS.....	10
ÍNDICE DE TABLAS.....	13
INTRODUCCIÓN	15
CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA	17
1.1 Problemática de ciberseguridad	17
1.2 Ciberseguridad en startups tecnológicas	19
1.3 Problemática del caso de estudio	20
1.4 Objetivo General	21
1.4.1 Objetivos específicos	21
1.4.2 Límites	21
1.4.3 Contexto.....	21
1.4.4 Valor de la memoria.....	22
CAPÍTULO 2: MARCO CONCEPTUAL	23
2.1 Introducción al caso de estudio	23
2.1.1 Problemática de desperdicio de alimentos	23
2.1.2 Causas del desperdicio en el hogar	24
2.1.3 Not Waste	25
2.1.4 Arquitectura de la solución.....	26
2.1.5 Prototipo	27

2.2 Seguridad informática	28
2.2.1 Que es la seguridad informática	28
2.3 Enfoques metodológicos de seguridad informática	29
2.3.1 Introducción a los enfoques metodológicos de seguridad informática	29
2.3.2 InfoSec Program.....	30
2.3.3 Cybersecurity Program	32
2.3.4 AppSec Program.....	33
2.3.5 SSDLC.....	34
2.3.6 Estándares de Verificación.....	38
CAPÍTULO 3: PROPUESTA DE SOLUCIÓN	40
3.1 Propuesta de solución	40
3.2 Justificación de la Metodología.....	40
3.3 OWASP SAMM.....	42
3.4 Como implementar OWASP SAMM	45
3.4.1 Pasos de implementación	46
3.4.2 Preparación	46
3.4.3 Evaluación	46
3.4.4 Establecer el objetivo.....	52
3.4.5 Definir el plan.....	54
3.5 Planificación de implementación en Not Waste	54
3.5.1 Preparación y análisis de contexto	55
3.5.2 Evaluación frente al modelo	56
3.5.3 Establecer objetivos.....	58
3.5.5 Planificar la implementación	59

CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN	61
4.1 Introducción al capítulo	61
4.2 Práctica SA-AD-2.....	61
4.2.1 Identificación de servicios de seguridad.....	61
4.2.2 Categorización de servicios.....	68
4.2.3 Selección de servicios	69
4.2.4 Detalle de servicios	70
4.2.5 Implementación	73
4.3 Práctica RT-CV-1	93
4.3.1 Proceso.....	94
4.3.2 Conocimiento previo.....	96
4.3.3 Pruebas de software	96
4.3.4 Herramientas	98
4.3.5 Implementación	101
4.3.6 Resumen de práctica	120
4.4 Resultados	121
4.4.1 Resultados práctica SA-AD-2.....	121
4.4.2 Resultados práctica RT-CV-1	123
CAPÍTULO 5: CONCLUSIONES	125
5.1 Cumplimiento Objetivo General	125
5.2 Limitaciones del Modelo aplicado	126
5.3 Recomendaciones	128
5.3.1 Importancia de pasos previos.....	128
5.3.2 Herramientas recomendadas	130

5.3.3 Recomendación a las startup.....	131
5.4 Trabajo Futuro.....	132
5.4.1 Mejoras en la calidad	132
5.5 Conclusión	132
REFERENCIAS BIBLIOGRÁFICAS.....	133

ÍNDICE DE FIGURAS

Figura 1: Estado de la ciber resiliencia 2024 Fuente: Global Cybersecurity Outlook, WEF 2024	18
Figura 2: ¿Tienes las habilidades necesarias para alcanzar tus objetivos de ciberseguridad? Fuente: Global Cybersecurity Outlook, WEF 2024	19
Figura 3: Causas del desperdicio de alimentos Fuente: Elaboración propia.....	24
Figura 4: Soluciones de Not Waste Fuente: Elaboración propia.....	25
Figura 5: Arquitectura de Not Waste Fuente Elaboración propia.....	26
Figura 6: Prototipo de Not Waste Fuente: Elaboración propia.....	27
Figura 7: Ontología de marcos de seguridad Fuente: Adaptado de Security Framework Ontology, Codific 2024	30
Figura 8: Procedimientos que definen el proceso de ciclo de vida de desarrollo de seguridad (SDL) Fuente: Informe de progreso del proceso SDL, Microsoft 2011.	36
Figura 9: Seguridad en el ciclo de vida del desarrollo de software (SSDL). Fuente: Ciclo de vida de desarrollo de software seguro (SSDLC), Snyk 2024.	36
Figura 10: Metodología DevSecOps Fuente: DevSecOps-Tools, Atlassian.....	37
Figura 11: Modelo de madurez SAMM Fuente: OWASP SAMM	43
Figura 12: Estructura de SAMM Fuente: OWASP SAMM	43
Figura 13: Criterios de calidad Fuente: Adaptado de OWASP SAMM, 2024.....	44
Figura 14: Metodología de implementación de SAMM Fuente: Adaptado de Guía de Inicio Rápido, por OWASP SAMM, 2024	45
Figura 15: Contenido caja de herramientas, OWASP SAMM Fuente: Elaboración propia ..	48
Figura 16: Estructura entrevista, autoevaluación SAMM Fuente: Elaboración propia.....	49
Figura 17: Tarjeta de puntuación "Scorecard", OWASP SAMM Fuente: Elaboración propia	51
Figura 18: Resultado generales SAMM Benchmark. Fuente: Elaboración propia.	53

Figura 19: Evaluación de madurez en seguridad API (Not Waste) Fuente: Elaboración propia.....	56
Figura 20: Puntuación de evaluación inicial de Not Waste Fuente: Elaboración propia	57
Figura 21: Cronograma de actividades SAMM. Fuente: Elaboración propia.	60
Figura 22: Conceptos previos a la identificación de servicios de seguridad. Fuente: Elaboración propia.....	62
Figura 23: Estructura JWT. Fuente: Elaboración propia.	64
Figura 24: Proxy inverso. Fuente: Cloudflare. (2024). ¿Qué es un Proxy inverso?	67
Figura 25: Firewall de aplicaciones. Fuente: Elaboración propia.	67
Figura 26: Diagrama general de servicios utilizados. Fuente: Elaboración propia.	70
Figura 27: Uptime de Better Stack. Fuente: Elaboración propia.....	72
Figura 28: Logtail de Better Stack. Fuente: Elaboración propia.	72
Figura 29: Comparación de infraestructura, previa y actual. Fuente: Elaboración propia. .	73
Figura 30: Comunicación entre cliente y API con seguridad aplicada Fuente: <i>Solution Overview</i> (Mobile Apps + API), Auth0 2024	76
Figura 31: Configuraciones importantes de autenticación mediante JWT en Django Fuente: Elaboración propia.....	77
Figura 32: Configuración de Flutter con Auth0. Fuente: Elaboración propia.	78
Figura 33: Registros DNS en Cloudflare. Fuente: Elaboración propia.	81
Figura 34: Comunicación cliente servidor predeterminado de Cloudflare Fuente: Elaboración propia.....	82
Figura 35: Comunicación cliente servidor con Cloudflare, modo cifrado completo. Fuente: Elaboración propia.....	82
Figura 36: Reglas de acceso al servidor mediante Cloudflare. Fuente: Elaboración propia.	83
Figura 37: Reglas personalizadas, WAF Cloudflare Fuente: Elaboración propia.....	84
Figura 38: Mitigación regla personalizada Header required, Cloudflare WAF Fuente: Elaboración propia.....	84

Figura 39: Mitigación regla predeterminada Modo bot figth, Cloudflare WAF Fuente: Elaboración propia.....	85
Figura 40: Ejemplo registros con Django. Fuente: Elaboración propia.	89
Figura 41: Registros de API Django. Fuente: Elaboración propia.	92
Figura 42: Registros API Django en Better Stack. Fuente: Elaboración propia.	92
Figura 43: Proceso Creación de pruebas en API Fuente: Elaboración propia.	94
Figura 44: Ejemplo código Pytest. Fuente: Elaboración propia.	100
Figura 45: Prueba AC-1. Fuente: Elaboración propia.	105
Figura 46: Validación AC-1. Fuente: Elaboración propia.	106
Figura 47: Verificación IV-1. Fuente: Elaboración propia.....	108
Figura 48: Números válidos. Fuente Elaboración propia.	110
Figura 49: Prueba IV-5. Fuente: Elaboración propia.	110
Figura 50: Prueba CF-1 Fuente: Elaboración propia.....	114
Figura 51: Verificación CF-2. Fuente: Elaboración propia.	115
Figura 52: Prueba CF-3 parte 1. Fuente: Elaboración propia.	116
Figura 53: Prueba CF-3 parte 2. Fuente: Elaboración propia.	117
Figura 54: Prueba CF-4. Fuente: Elaboración propia.....	119
Figura 55: Prueba CF-5, Postman. Fuente: Elaboración propia.	120

ÍNDICE DE TABLAS

Tabla 1: Comparación de marcos de seguridad informática Fuente: Elaboración propia...	41
Tabla 2: Como responder a las preguntas de SAMM. Fuente: Elaboración propia.	50
Tabla 3: Ejemplo de pregunta y criterios de calidad. Fuente: elaboración propia.	50
Tabla 4: Ejemplo respuestas de actividad. Fuente: Elaboración propia.	50
Tabla 5: Equipo de Not Waste Fuente: Elaboración propia	55
Tabla 6: Objetivos de implementación Fuente: Elaboración propia	58
Tabla 7: Plan de actividades SAMM. Fuente: Elaboración propia.....	59
Tabla 8: Categorización de servicios de seguridad. Fuente: Elaboración propia.	68
Tabla 9: Selección de servicios de seguridad. Fuente: Elaboración propia.....	69
Tabla 10: Alcance de monitoreo. Fuente: Elaboración propia.....	88
Tabla 11: Formato de logs para API Dango. Fuente: Elaboración propia.	91
Tabla 12: Tipos de prueba. Fuente: Elaboración propia.	96
Tabla 13: Diferencia entre control de seguridad y caso de prueba Fuente: Elaboración propia.....	98
Tabla 14: Framework de pruebas por Lenguaje de programación. Fuente: Elaboración propia.....	99
Tabla 15: Detalle de entorno de prueba. Fuente: Elaboración propia.	103
Tabla 16: Caso de prueba AC-1. Fuente: Elaboración propia.....	104
Tabla 17: Caso de prueba AC-7. Fuente: Elaboración propia.....	105
Tabla 18: Caso de prueba AC-8. Fuente: Elaboración propia.....	106
Tabla 19: Caso de prueba AC-9. Fuente: Elaboración propia.....	107
Tabla 20: Caso de prueba IV-1. Fuente: Elaboración propia.	107
Tabla 21: Caso de prueba IV-5. Fuente: Elaboración propia.	108
Tabla 22: Caso de prueba IV-8. Fuente: Elaboración propia.	110

Tabla 23: Caso de prueba FR-1. Fuente: Elaboración propia.	112
Tabla 24: Caso de prueba CF-1. Fuente: Elaboración propia.	113
Tabla 25: Caso de prueba CF-2. Fuente: Elaboración propia.	114
Tabla 26: Caso de prueba CF-3. Fuente: Elaboración propia.	115
Tabla 27: Caso de prueba CF-4. Fuente: Elaboración propia.	118
Tabla 28: Caso de prueba CF-5. Fuente: Elaboración propia.	119
Tabla 29: Evaluación práctica SA-AD-2. Fuente: Elaboración propia.	121
Tabla 30: Evaluación práctica RT-CV-1. Fuente: Elaboración propia.	123

INTRODUCCIÓN

En el mundo dinámico y competitivo de las startups, el enfoque principal suele estar en desarrollar productos innovadores, atraer clientes y garantizar un rápido crecimiento. Sin embargo, a medida que estas empresas emergentes comienzan a consolidar su posición en el mercado, la ciberseguridad se convierte en una preocupación cada vez más apremiante. Implementar medidas de seguridad robustas desde el principio es crucial para proteger los datos de los usuarios, mantener la confianza de los clientes, así como evitar riesgos legales y financieros.

El desafío radica en una serie de obstáculos que afectan a todas las organizaciones, tal como lo destaca el informe *Global Cybersecurity Outlook (GCO) 2024* del Foro Económico Mundial (WEF). Entre estos desafíos se encuentra "el creciente costo de acceso a servicios cibernéticos innovadores, herramientas, habilidades y experiencia", un problema que impacta especialmente a las pequeñas empresas debido a sus recursos limitados. Estas empresas enfrentan dificultades tanto para competir por el talento especializado como para adquirir las herramientas necesarias. Además, su limitada experiencia en normativas y estándares de seguridad puede hacer que la tarea de proteger un proyecto de software resulte desalentadora y compleja. La combinación de equipos pequeños, presupuestos ajustados y plazos estrictos suele llevar a priorizar la funcionalidad sobre la seguridad, dejando vulnerabilidades que los atacantes pueden explotar.

A pesar de las dificultades, integrar la ciberseguridad en el ciclo de vida del desarrollo de software no solo es posible, sino también esencial, metodologías como OWASP SAMM (Software Assurance Maturity Model) ofrecen a las startups una base sólida para crecer de manera segura y sostenible.

En este contexto, se tomó como caso de estudio a Not Waste, una startup que enfrenta el desafío de seleccionar un enfoque metodológico para garantizar la seguridad a las partes interesadas del proyecto: clientes, inversores y la propia organización.

En los capítulos siguientes, se analizan dos problemáticas principales: por un lado, el desperdicio de alimentos, que constituye el enfoque central de Not Waste, y por otro, la amenaza de la ciberdelincuencia, un desafío común para las startups y las organizaciones de todo el mundo.

Para abordar esta última problemática, se adopta un marco de ciberseguridad y su metodología de aplicación, seleccionando entre diversas alternativas como ISO 27001, BSIMM, NIST CSF, entre otras. El objetivo de este trabajo es proporcionar una guía para startups tecnológicas que busquen establecer una postura de seguridad estructurada y escalable.

CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA

1.1 Problemática de ciberseguridad

En el contexto actual, la seguridad informática se ha convertido en una prioridad crucial debido al incremento significativo de las amenazas cibernéticas. Para enfrentar los riesgos, es fundamental que las organizaciones desarrollen e implementen estrategias de protección efectivas para salvaguardar su información y operaciones.

La falta de una adecuada protección puede acarrear consecuencias graves, tales como pérdidas operativas, financieras y legales. El informe *Data Breach Report* (IBM, 2024), que analizó a 604 organizaciones afectadas por brechas de datos entre marzo de 2023 y febrero de 2024, señaló que el costo promedio de una violación de datos en 2024 alcanzó los 4,88 millones de dólares, lo que representa un aumento del 10% respecto al año anterior. Este incremento se atribuye principalmente al aumento de los costos asociados con la pérdida de negocio, como el tiempo de inactividad operativo, la pérdida de clientes y los gastos derivados de las respuestas posteriores a las vulneraciones.

Frente a esta creciente necesidad de fortalecer la seguridad informática para evitar interrupciones en las operaciones y proteger los datos personales de los usuarios, la ciber resiliencia se ha convertido en un concepto clave, esta se define, según un artículo de IBM, como la “capacidad de una organización para prevenir, resistir y recuperarse de incidentes de ciberseguridad” (IBM, 2024).

Un análisis reciente presentado en el informe Global Cybersecurity Outlook (GCO) 2024 (World Economic Forum, 2024) destaca una creciente brecha entre las organizaciones que han alcanzado un nivel avanzado de ciber resiliencia y aquellas que enfrentan dificultades significativas para mantenerse a flote. Esta brecha se refleja en la disminución de un “grupo intermedio” de organizaciones que apenas cumplen con los estándares mínimos de ciber resiliencia. El informe señala que “las grandes organizaciones han demostrado avances notables en ciber resiliencia, mientras que las pymes¹ mostraron un descenso significativo”.

Diversos factores contribuyen a ampliar la brecha entre aquellas organizaciones capaces de cumplir con sus objetivos de seguridad y las que no pueden. El informe del Foro Económico Mundial destaca una serie de obstáculos entre los que se encuentran el creciente costo de acceso a servicios cibernéticos innovadores, herramientas, habilidades y experiencia, las

¹ El diccionario de la lengua española define pyme a pequeñas y medianas empresas.

cuales siguen influyendo en la capacidad del ecosistema global para construir un ciberespacio más seguro.

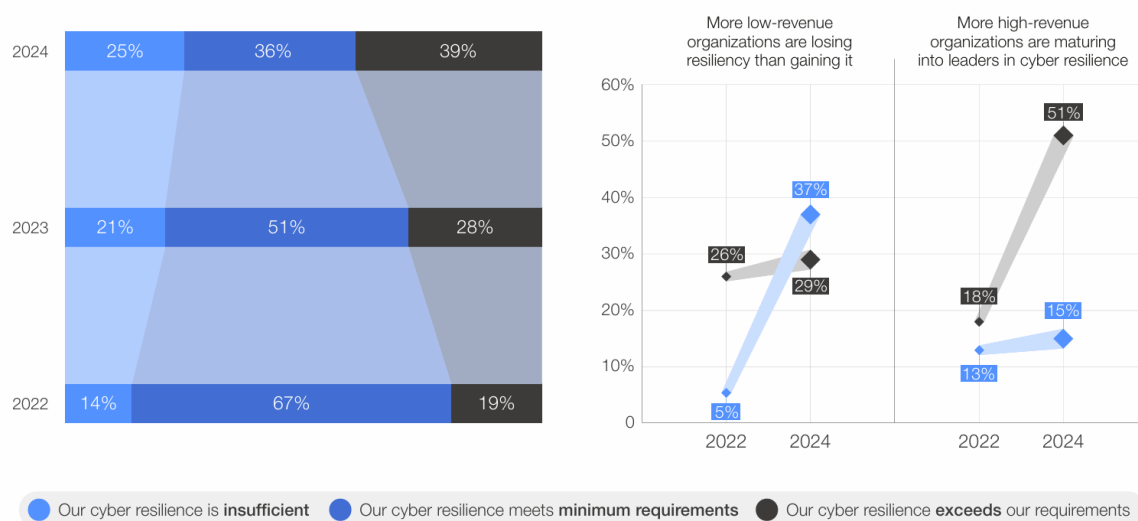


Figura 1: Estado de la ciber resiliencia 2024
Fuente: Global Cybersecurity Outlook, WEF 2024

De acuerdo con el gráfico presentado en la figura 1, las organizaciones han experimentado cambios significativos en su capacidad para enfrentar amenazas cibernéticas durante los últimos tres años:

- **Resiliencia insuficiente:** El porcentaje de organizaciones que consideran insuficiente su resiliencia aumentó del 14% en 2022 al 25% en 2024, reflejando un deterioro en ciertos sectores.
- **Resiliencia mínima:** La proporción de organizaciones que cumplen con los requisitos básicos cayó del 67% en 2022 al 36% en 2024, lo que indica que muchas organizaciones están transitando hacia extremos: o pierden resiliencia o la fortalecen significativamente.
- **Resiliencia avanzada:** El porcentaje de organizaciones que superan los requisitos de resiliencia creció del 19% en 2022 al 39% en 2024, demostrando una mejora en aquellas con mayores recursos e inversiones en ciberseguridad.

En el segundo gráfico de la figura 1, se muestra que un mayor número de empresas con menores ingresos están perdiendo ciber resiliencia en lugar de ganarla. En 2024, el 37% de las empresas con menores ingresos presentaron niveles insuficientes de ciber-resiliencia en contraste al 2022 que solo representaban el 5%.

1.2 Ciberseguridad en startups tecnológicas

Las habilidades y el conocimiento de ciberseguridad forman un papel clave en la seguridad en las organizaciones. El reporte Fortinet *Cybersecurity Skills Gap 2022* (Fortinet, 2022) detalla que “el 80% organizaciones sufrieron una o más brechas de seguridad que pudieron atribuir a la falta de habilidades y/o conciencia en ciberseguridad”, y está siendo particularmente difícil para las organizaciones encontrar las habilidades necesarias para enfrentar los riesgos y amenazas de la seguridad informática algunos, el reporte de Fortinet señala que 60% de las organizaciones tiene dificultades para reclutar talento en ciberseguridad, y el 52% tiene problemas para retenerlo.

La dificultad para encontrar profesionales con las habilidades y conocimientos necesarios en ciberseguridad, junto con los desafíos identificados por el Global Cybersecurity Outlook (WEF, 2024), como el creciente costo de acceso a servicios cibernéticos innovadores, herramientas, habilidades y experiencia, afecta especialmente a las pequeñas empresas.

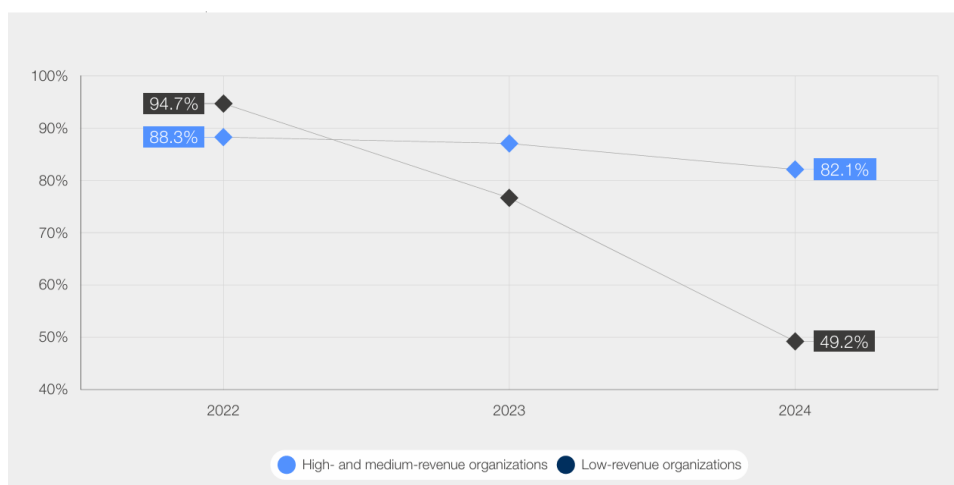


Figura 2: ¿Tienes las habilidades necesarias para alcanzar tus objetivos de ciberseguridad?

Fuente: Global Cybersecurity Outlook, WEF 2024

Las limitaciones presupuestarias de las startups y pequeñas empresas pueden dificultar la inversión en herramientas avanzadas de ciberseguridad, la contratación de personal especializado o la implementación de prácticas robustas de protección². En el caso de las startups, su naturaleza de reciente creación y su enfoque en el crecimiento acelerado les hace aún más complicado competir por talento calificado en ciberseguridad frente a empresas consolidadas con mayores recursos financieros y una gestión organizacional más avanzada. En estos entornos, donde los recursos son limitados y la prioridad recae en la rápida creación de soluciones tecnológicas, la seguridad suele relegarse a un segundo plano, exponiéndolas a mayores riesgos cibernéticos.

1.3 Problemática del caso de estudio

Una de las principales dificultades que enfrentan algunas organizaciones al intentar protegerse de amenazas cibernéticas radica en responder preguntas clave: ¿qué activos necesitan protección?, ¿cómo deben protegerse?, ¿qué herramientas deben utilizarse? y ¿cómo verificar si las medidas implementadas son efectivas? Para abordar estos desafíos, es fundamental contar con un plan, una estrategia y una metodología que guíe la aplicación de dichas estrategias. Estas interrogantes destacan la importancia de disponer de “guías” que orienten la gestión efectiva de la ciberseguridad.

Este es el caso de *Not Waste*: ¿cómo puede la organización proteger sus activos, incluyendo la aplicación y la información tanto de la empresa como de sus usuarios? ¿Cómo prepararse para lograr una adecuada ciber resiliencia? Para responder estas preguntas, resulta indispensable utilizar un marco de ciberseguridad que proporcione un enfoque estructurado y eficaz.

Según un artículo de Splunk escrito por Chrissy Kidd (2024) un marco de ciberseguridad es, “En esencia, un conjunto estructurado de mejores prácticas, estándares y pautas diseñadas para ayudar a las empresas a gestionar los riesgos de ciberseguridad. Ayuda a las organizaciones a identificar, evaluar y gestionar los riesgos potenciales y a proteger sus activos digitales”. La ventaja de estos marcos es que ofrecen una guía clara para proteger los activos digitales y establecer un estándar de referencia confiable de las estrategias de seguridad. Su implementación ayuda a responder de manera más precisa a las preguntas iniciales, proporcionando un enfoque estructurado frente a la complejidad de entorno digital.

² Prácticas específicas como sistemas de información y gestión de eventos de seguridad (SIEM), adopción de estándares como el ISO 27001 o cumplimiento regulatorio como GDPR

Dentro de los marcos de la ciberseguridad las estrategias más utilizadas son ISO/IEC 27001, NIST CSF, BSIMM, SAMM, CIS Control, COBIT, Microsoft SDL, SAFECode. No obstante, la amplia variedad de opciones puede resultar abrumadora para organizaciones emergentes que, además de enfrentar recursos limitados y la presión por priorizar el crecimiento, carecen de la experiencia necesaria para determinar qué enfoque es el más adecuado para protegerse ante las diversas amenazas. Sin una orientación clara, estas decisiones críticas pueden convertirse en una barrera significativa para garantizar la protección de sus activos digitales y la continuidad de sus operaciones.

1.4 Objetivo General

Implementar prácticas de seguridad informática utilizando una estrategia basada en el modelo de madurez OWASP SAMM y su metodología de aplicación, con el propósito de reforzar la seguridad de la aplicación Not Waste y proteger sus activos frente a posibles amenazas.

1.4.1 Objetivos específicos

- Seleccionar un enfoque metodológico de seguridad estructurado y factible para startups tecnológicas con bajos recursos.
- Desarrollar un plan de seguridad basado en OWASP SAMM, que incluye implementación y evaluación de prácticas de seguridad.
- Documentar el proceso de implementación utilizado en Not Waste, para servir como caso de uso para otras startups tecnológicas.

1.4.2 Límites

- Esta memoria de título no comprende la realización de todos los puntos que engloba el marco de seguridad seleccionado, el proceso de seguridad de una organización no es un objetivo con final, sino que se desarrolla y aplica en toda la vida de la organización.
- La implementación de actividades está descritas e implementadas de manera simplificada.

1.4.3 Contexto

Se replicó un escenario real de desarrollo de software para aprender y aplicar medidas de seguridad, utilizando un enfoque estructurado apropiado a las startup tecnológicas con bajos recursos, en algunos casos sin profesionales de seguridad informática.

Se simuló un contexto empresarial en el que una startup tecnológica está construyendo un sistema basado en tecnologías como Django, Flutter, AWS, entre otras.

1.4.4 Valor de la memoria

Este proyecto aporta valor al generar conocimiento sobre cómo utilizar un enfoque metodológico de seguridad informática para guiar la implementación de prácticas de seguridad en proyectos de software liderados por startups tecnológicas.

Desde una perspectiva académica, el proyecto sirve como un aporte al conocimiento sobre ciberseguridad y como base para otros estudiantes, profesionales o startups tecnológicas que busquen mejorar sus habilidades en este campo.

CAPÍTULO 2: MARCO CONCEPTUAL

2.1 Introducción al caso de estudio

En este capítulo se presenta el caso de estudio de una organización que ha desarrollado una aplicación móvil innovadora llamada Not Waste. Esta aplicación está diseñada para abordar una problemática global de gran relevancia: el desperdicio de alimentos. A través de Not Waste, los usuarios pueden gestionar de manera eficiente sus alimentos, reduciendo así las pérdidas por caducidad o mal manejo.

El caso de estudio tiene como objetivo demostrar la aplicación de un marco metodológico de ciberseguridad en el desarrollo y operación de esta aplicación. En el proceso, se analizarán aspectos clave de la seguridad en el ciclo de vida del software, asegurando que Not Waste cumpla con prácticas relevantes para proteger la información de sus usuarios y garantizar la continuidad del servicio.

2.1.1 Problemática de desperdicio de alimentos

La pérdida y el desperdicio de alimentos (PDA) representan un problema crítico con graves implicancias como las pérdidas económicas, el mal uso de recursos naturales y la increíble paradoja de que mientras millones de personas padecen hambre, en otros sectores se desperdician toneladas de alimentos.

Es tal la importancia de este problema que forma parte de uno de los objetivos del desarrollo sostenible de la ONU el ODS 12 “Garantizar modalidades de consumo y producción sostenibles” donde establece como una de sus metas (OSD 12.3) reducir a la mitad el desperdicio de alimentos per cápita mundial en la venta al por menor y a nivel de los consumidores y reducir las pérdidas de alimentos en las cadenas de producción y suministro para 2030.

Según un reporte de la ONU (2024), en 2022 se generaron 1.050 millones de toneladas de desperdicios alimentarios, incluyendo restos no comestibles. De este total, el 60% provino de los hogares, el 28% de proveedores de servicios alimentarios y el 12% del comercio minorista. Estos datos evidencian que una gran parte del desperdicio ocurre en la etapa final de la cadena alimentaria, donde los consumidores, ya sea por falta de conocimiento o de organización, descartan alimentos que podrían haber sido consumidos o redistribuidos.

2.1.2 Causas del desperdicio en el hogar

En base al estudio nacional de Maggi e Ipsos (2023) sobre el desperdicio de alimentos en Chile donde se encuestaron a más de 600 personas, se identificaron los distintos momentos que pueden afectar al desperdicio de alimentos en el hogar, los cuales son compra, almacenamiento y preparación.

Del estudio mencionado se sintetizaron las siguientes causas:

Compra:

- No tiene una visión clara de los alimentos en casa.
- Compra más de lo necesario o productos que ya tiene

Almacenamiento:

- Las personas no saben cómo almacenar ciertos alimentos

Preparación:

- Olvida usar productos con fecha próxima de caducidad y prefiere alimentos más visibles o fáciles de usar.
- Sobran alimentos en el plato que luego se desperdician.



Figura 3: Causas del desperdicio de alimentos

Fuente: Elaboración propia

2.1.3 Not Waste

Not Waste es una aplicación móvil creada por una organización de estudiantes de la Universidad Técnica Federico Santa María (UTFSM), con el objetivo de reducir el desperdicio de alimentos de manera simple. Su enfoque se centra específicamente en el desperdicio proveniente del sector doméstico.

En base a las causas del desperdicio de alimentos planteadas anteriormente, se desarrolló esta aplicación, la cual ofrece soluciones en distintos momentos donde se produce el desperdicio alimenticio en el sector doméstico. Estas soluciones se ilustran en la figura 4.



Figura 4: Soluciones de Not Waste

Fuente: Elaboración propia

2.1.4 Arquitectura de la solución

Se presenta la arquitectura tecnológica de la solución, que consta de una aplicación móvil escrita en Flutter la cual se conecta API en Django para acceder a los datos almacenados en una base de datos de PostgreSQL

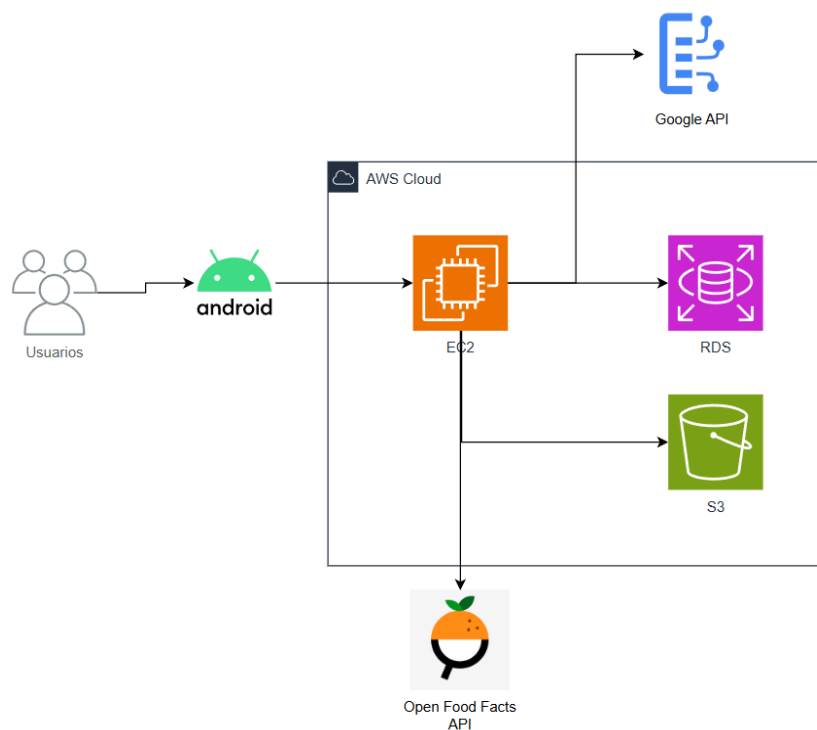


Figura 5: Arquitectura de Not Waste
Fuente Elaboración propia

2.1.5 Prototipo

En esta sección se presentará parte del prototipo de Not Waste, desarrollado para facilitar la gestión de alimentos y de esa manera evitar el reducir el desperdicio de alimentos en el hogar, acá una breve descripción de las pantallas expuestas en la figura 6:

- **Pantalla 1:** La primera pantalla es la *home screen*, donde aparecen los alimentos a punto de vencer. Además, se muestran 3 funcionalidades para ingresar de manera rápida los productos al inventario, las cuales son: tomar una foto de una boleta, tomar una foto de un alimento y escanear el código de barra del alimento.
- **Pantalla 2:** La segunda pantalla muestra un inventario de los alimentos, con detalles de los productos como su cantidad y los días que quedan para su fecha de vencimiento.
- **Pantalla 3:** La tercera pantalla muestra estadísticas del consumo de los alimentos, como las categorías de los alimentos y la cantidad que ha sido desperdiciada y consumida en cada mes.

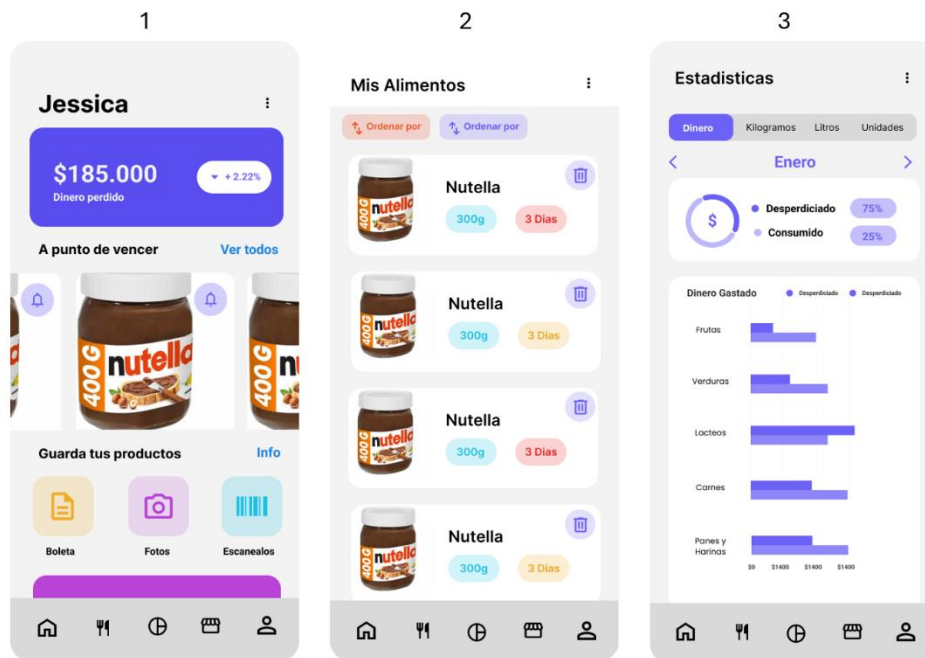


Figura 6: Prototipo de Not Waste
Fuente: Elaboración propia

2.2 Seguridad informática

2.2.1 Que es la seguridad informática

La seguridad informática también conocida como “ciberseguridad es la práctica de proteger equipos, redes, aplicaciones de software, sistemas críticos y datos de posibles amenazas digitales” (AWS, 2024). Tiene como objetivo establecer normas y medidas que minimicen los riesgos a la información y la infraestructura informática.

Según el libro CompTIA Security+ (Chapple & Seidl, 2020) un programa de seguridad informática está basado en tres pilares fundamentales:

- La confidencialidad se enfoca en impedir que individuos sin autorización puedan acceder a datos privados o sensibles.
- La integridad busca mantener los datos y sistemas libres de alteraciones no permitidas, ya sea de forma accidental o deliberada.
- La disponibilidad se asegura de que los sistemas y la información estén operativos y accesibles para los usuarios autorizados en el momento que los requieran.

2.3 Enfoques metodológicos de seguridad informática

2.3.1 Introducción a los enfoques metodológicos de seguridad informática

Existen múltiples estrategias para implementar la seguridad informática, entre las cuales destacan la ISO/IEC 27001, NIST CSF, SAMM y BSIMM, entre otras. Estas estrategias suelen clasificarse de diferentes formas, como marcos de trabajo o estándares de seguridad, según diversos autores. En este trabajo, se utilizó una clasificación en capas basada en las "áreas" de protección que abarca cada estrategia de seguridad, siguiendo el enfoque propuesto por el Doctor en Ciberseguridad Aram Hovsepyan.

Como explica Hovsepyan de Codific (2024), una manera simple de categorizar las diferentes estrategias mencionadas serían las siguientes:

- **Ciclo de vida de desarrollo de software seguro:** el SSLC es un marco de procesos que incorpora prácticas y controles de seguridad en cada etapa del desarrollo de software para garantizar la seguridad de los códigos y los sistemas. Ejemplos: Microsoft SDL, SAFECode.
- **Seguridad de aplicaciones:** Un programa de seguridad de aplicaciones es un enfoque estructurado para gestionar y mejorar la seguridad de las aplicaciones a lo largo de su ciclo de vida mediante la integración de prácticas de seguridad en el ciclo de vida del desarrollo de software (SDLC). Ejemplos: NIST SSDF, OWASP SAMM, BSIMM.
- **Ciberseguridad:** Un programa de ciberseguridad es una estrategia integral para gestionar y mitigar los riesgos de seguridad en toda la infraestructura digital de una organización, incluidas las redes, los sistemas y las aplicaciones. Ejemplos: NIST CSF, Fundamentos de ciberseguridad.
- **Seguridad de la información:** Un programa de seguridad de la información es una iniciativa amplia para proteger los activos de información de una organización, tanto digitales como físicos, a través de políticas, controles y prácticas de gestión de riesgos. Ejemplos: ISO 27001, COBIT.

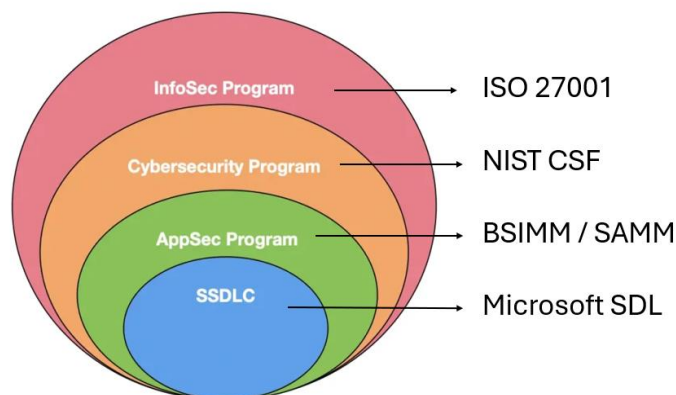


Figura 7: Ontología de marcos de seguridad

Fuente: Adaptado de Security Framework Ontology, Codific 2024

Como se muestra en la figura 7 algunos enfoques abarcan dentro de sus estrategias otros enfoques de seguridad informática. Sin embargo, es necesario explicar que no todos estos marcos son aplicables a todas las organizaciones, su implementación depende de diversos factores como el sector o el tamaño de la empresa, tomemos el ejemplo de una pequeña empresa dedicada a la publicidad e imprenta, es posible que esta organización no desarrolle sus propios sistemas informáticos, por lo que no tendrá que preocuparse por asegurar el ciclo de vida del software (SSDLC), pero sí puede implementar un programa de ciberseguridad como el NIST CSF el cual tiene un enfoque más amplio.

2.3.2 InfoSec Program

La seguridad de la información (InfoSec) es la protección de la información importante contra el acceso, la divulgación, el uso, la alteración o la interrupción no autorizados. Ayuda a garantizar que los datos confidenciales de la organización estén disponibles para los usuarios autorizados, permanezcan confidenciales y mantengan su integridad. (IBM, 2024)

2.3.2.1 ISO 27001

La estrategia más popular para implementar un programa de seguridad de la información es la norma ISO 27001, que pertenece a la familia de estándares ISO/IEC 27000. Esta serie de normas se centra en los Sistemas de Gestión de la Seguridad de la Información (SGSI). El propósito central de un SGSI es “proporcionar protección a la información sensible o de valor. La información sensible incluye información sobre los empleados, clientes y proveedores. La información de valor incluye propiedad intelectual, datos financieros, registros legales, datos comerciales y datos operativos” (NQA, 2013)

La norma ISO 27001 es desarrollada conjuntamente por la Organización Internacional de Normalización (ISO) y la Comisión Electrotécnica Internacional (IEC). Proporciona un marco

para la gestión de la seguridad de la información y abarca varios aspectos, como la protección de datos, la gestión de riesgos y la implementación de controles de seguridad.

El anexo A de la ISO/IEC 27001 incluye 114 controles de seguridad de la información “basados en buenas prácticas, que deben ser considerados al desarrollar el plan de tratamiento de riesgos que establece la norma ISO 27001” (NQA, 2013), estos controles se detallan con más profundidad en la ISO/IEC 27002.

Los principales estándares de la familia ISO/IEC 27000 son los siguientes:

- ISO/IEC 27000:2018 - Visión general y vocabulario: Define la terminología utilizada en toda la serie de normas ISO/IEC 27000. Es el punto de referencia para entender los términos y conceptos clave.
- ISO/IEC 27001:2013 - Requisitos del SGSI: Especifica los requisitos para establecer, implementar, mantener y mejorar continuamente un SGSI. Es la norma principal para la certificación y gestión de la seguridad de la información.
- ISO/IEC 27002:2022 - Código de buenas prácticas para controles de seguridad de la información: Proporciona directrices y recomendaciones para implementar controles de seguridad de la información basados en las mejores prácticas. Es un complemento de la ISO/IEC 27001, ofreciendo detalles sobre cómo aplicar los controles mencionados en el Anexo A de la ISO/IEC 27001.

2.3.3 Cybersecurity Program

2.3.3.1 NIST Cybersecurity Framework (NIST CSF)

El NIST³ Cybersecurity Framework como se describe en la publicación *The NIST Cybersecurity Framework (CSF) 2.0* (NIST, 2024) es un conjunto de directrices y mejores prácticas diseñado para ayudar a las organizaciones a gestionar y reducir los riesgos de ciberseguridad. El marco se centra en la identificación y gestión de riesgos, la protección de los sistemas y la respuesta y recuperación ante incidentes. Esta organizado en cinco funciones principales:

- Identificar (Identify): Desarrollar una comprensión organizacional para gestionar el riesgo de ciberseguridad a los sistemas, personas, activos, datos y capacidades.
- Proteger (Protect): Desarrollar e implementar las salvaguardias apropiadas para asegurar los servicios críticos y limitar o contener el impacto de un evento de ciberseguridad.
- Detectar (Detect): Desarrollar e implementar las actividades de monitoreo para identificar la ocurrencia de un evento de ciberseguridad.
- Responder (Respond): Desarrollar e implementar las actividades apropiadas para tomar acción ante un evento de ciberseguridad detectado.
- Recuperar (Recover): Desarrollar e implementar planes para la resiliencia y la restauración de cualquier capacidad o servicio que se haya visto afectado por un evento de ciberseguridad.

³ Instituto Nacional de Normas y Tecnología (NIST), fue fundado en 1901 y actualmente forma parte del Departamento de Comercio de Estados Unidos. Ha creado una serie de publicaciones para la mejora de la seguridad cibernética e infraestructuras críticas, entre las cuales se encuentra NIST CSF, NIST 800-53, NIST 800-218.

2.3.4 AppSec Program

Como se explica en un artículo de Black Duck (2024), la seguridad de aplicaciones (AppSec) son los procesos, prácticas y herramientas que se utilizan para identificar, reparar y proteger contra vulnerabilidades en las aplicaciones, a lo largo del ciclo de vida del desarrollo de software (SDLC). La seguridad de aplicaciones implica una amplia gama de herramientas y metodologías, pero todas tienen el mismo objetivo: identificar debilidades y vulnerabilidades y solucionarlas antes de que puedan ser explotadas.

Para la realización de los programas de seguridad de aplicaciones, son muy útiles los modelos de madurez de seguridad, ya que mejoran los procesos de seguridad conforme al nivel actual que poseen.

2.3.4.1 Modelo de madurez

Los modelos de madurez son importantes en un AppSec Program, ya que ayudan al software a mejorar de manera progresiva en los ámbitos de seguridad de la aplicación.

Los modelos de Modelo de Madurez de Capacidades para Software (CMM) fueron desarrollados por la Universidad Carnegie-Mellon para el *Software Engineering Institute* (SEI).

El libro *Spanish Language Translation of CMMI for Development, v1.3* (Carnegie Mellon University, 2010) describe que los CMMs “se centran en mejorar los procesos de una organización. Contienen los elementos esenciales de los procesos de una o más disciplinas y describen un camino evolutivo de mejora desde inmaduros a procesos disciplinados y maduros con calidad y eficacia mejoradas”.

Este tipo de modelo ayuda a las organizaciones a identificar su estado actual, establecer objetivos claros y trazar un camino para mejorar de manera progresiva. Los modelos de madurez generalmente están organizados en niveles jerárquicos, donde cada nivel representa un grado de desarrollo o sofisticación mayor que el anterior.

2.3.4.2 Modelos de Madurez de la seguridad del software

Respecto a la seguridad informática hay dos modelos de madurez populares, los cuales son BSIMM y SAMM, quienes tienen diferencias en sus enfoques los cuales se mencionan a continuación:

Marcos prescriptivos: Estos marcos indican cómo debería implementarse un SDLC seguro. Proporcionan un conjunto de directrices, pasos y mejores prácticas que las organizaciones pueden seguir para mejorar la seguridad del software desde su concepción hasta su implementación. Ejemplos:

- SAMM: Ofrece un modelo claro de actividades y prácticas para gestionar la seguridad en el SDLC. (OWASP, 2024)

Marcos descriptivos: Estos describen cómo se está aplicando en la práctica la seguridad en un SDLC. Se basan en estudios de casos reales y miden el estado actual de los procesos de seguridad en las organizaciones, identificando áreas de mejora. Ejemplo:

- BSIMM (Building Security In Maturity Model): Se basa en la observación de prácticas reales en varias empresas, mostrando qué están haciendo en cuanto a seguridad en el desarrollo de software. (Black Duck, 2023)

La diferencia fundamental entre AppSec (Seguridad de Aplicaciones) y los modelos de madurez como BSIMM (Building Security In Maturity Model) y SAMM (Software Assurance Maturity Model) radica en sus enfoques y objetivos dentro del campo de la seguridad en el desarrollo de software. AppSec se refiere a una disciplina integral que abarca un conjunto de prácticas, procesos y herramientas dirigidos a proteger las aplicaciones a lo largo de su ciclo de vida, desde su diseño hasta su implementación y mantenimiento, asegurando que sean resilientes frente a amenazas y vulnerabilidades. En contraste, BSIMM y SAMM son modelos de madurez que proporcionan marcos metodológicos específicos para evaluar el nivel de madurez de las prácticas de seguridad dentro de una organización, permitiendo identificar áreas de mejora y establecer planes para fortalecer las capacidades de seguridad de las aplicaciones. Mientras que AppSec es una práctica general centrada en la protección de las aplicaciones, los modelos BSIMM y SAMM sirven como guías estructuradas que ayudan a medir, comparar y mejorar continuamente la seguridad de las aplicaciones dentro de un contexto organizacional.

2.3.5 SSDLC

El **SSDLC** (Secure Software Development Life Cycle, por sus siglas en inglés) es un enfoque que integra prácticas de seguridad en cada fase del ciclo de vida del desarrollo de software. Su objetivo es identificar y mitigar los riesgos de seguridad desde las primeras etapas del desarrollo hasta la implementación y mantenimiento del software.

2.3.5.1 Ciclo de vida del software (SDLC)

Para comprender qué es la seguridad en el ciclo de vida del software (SSDLC), primero es necesario entender qué es el ciclo de vida del software (SDLC). Como se explica uno de los artículos de la empresa de software y tecnología Globant (2024) el SDLC es “un proceso estructurado utilizado para desarrollar aplicaciones de software. Consiste en varias fases: definición, diseño, desarrollo, pruebas, implementación y mantenimiento. Cada fase asegura que el software cumpla con los requisitos y funcione correctamente”.

Actualmente hay una variedad de formas en que distintas literaturas descomponen el ciclo de vida del software, por ejemplo, el artículo de Globant (2024) descompone el ciclo de vida del software en las siguientes fases:

- Fase de definición: Esta fase inicial implica la recopilación de requisitos, la definición de los objetivos del proyecto y la determinación de la viabilidad. Establece la base para todo el proyecto.
- Fase de diseño: En esta fase, se crea la arquitectura y el diseño del sistema basado en los requisitos recopilados. Esto incluye estructuras de datos, arquitectura de software e interfaces de usuario.
- Fase de desarrollo: Los desarrolladores escriben el código según las especificaciones del diseño. Aquí es donde el software comienza a tomar forma.
- Fase de pruebas: El software se somete a pruebas rigurosas para identificar y corregir errores, asegurando que cumpla con los estándares de calidad y funcione como se espera.
- Fase de implementación: El producto final se implementa para los usuarios finales. Esta fase también puede incluir capacitación de usuarios y documentación.
- Fase de mantenimiento: Se proporciona soporte continuo y actualizaciones para corregir problemas, mejorar el rendimiento y añadir nuevas características según sea necesario.

Los modelos SDLC

Un modelo de ciclo de vida del desarrollo de software (SDLC) organiza conceptualmente el proceso para facilitar su implementación. Existen diferentes modelos que ordenan las fases del SDLC en diversos ordenes cronológicos. (AWS, 2024) Los modelos más conocidos son: Cascada, Iterativo, Espiral, Ágil.

2.3.5.2 Seguridad en el ciclo de vida del software

En los años 2000, organizaciones como Microsoft popularizaron el concepto de SDL (Security Development Lifecycle), integrando buenas prácticas de seguridad en cada fase del desarrollo de software



Figura 8: Procedimientos que definen el proceso de ciclo de vida de desarrollo de seguridad (SDL)

Fuente: Informe de progreso del proceso SDL, Microsoft 2011.

En la actualidad con las nuevas metodologías del desarrollo de software como las metodologías ágiles, han hecho que la seguridad en el ciclo de vida del software también evolucione, pero sigue teniendo el mismo objetivo la “introducción de seguridad y privacidad en todas las fases del proceso de desarrollo” (Microsoft, 2011).

Entre las practicas actuales más comunes de la seguridad en el ciclo de vida del software se encuentran, la evaluación de riesgos, modelado de amenazas, análisis estático del código y pruebas de seguridad. No obstante, cada organización establece sus propias prácticas para el desarrollo seguro.

Secure Software Development Life Cycle (SSDLC)



Figura 9: Seguridad en el ciclo de vida del desarrollo de software (SSDL).

Fuente: Ciclo de vida de desarrollo de software seguro (SSDLC), Snyk 2024.

2.3.5.3 DevSecOps

“DevSecOps es una práctica de seguridad de aplicaciones (AppSec) que introduce la seguridad en las primeras etapas del ciclo de vida del desarrollo de software (SDLC)” (Black Duck, 2024) Según Black Duck (2024) La integración continua de la seguridad en todas las fases del ciclo de vida del desarrollo de software permite a los equipos de DevOps entregar aplicaciones con mayor rapidez y calidad, asegurando su protección. Al incluir la seguridad desde el inicio del proceso, se pueden identificar y corregir vulnerabilidades de manera temprana. Este enfoque, denominado "desplazamiento a la izquierda", traslada las pruebas de seguridad al ámbito de los desarrolladores, permitiéndoles resolver problemas directamente en su código durante el desarrollo.

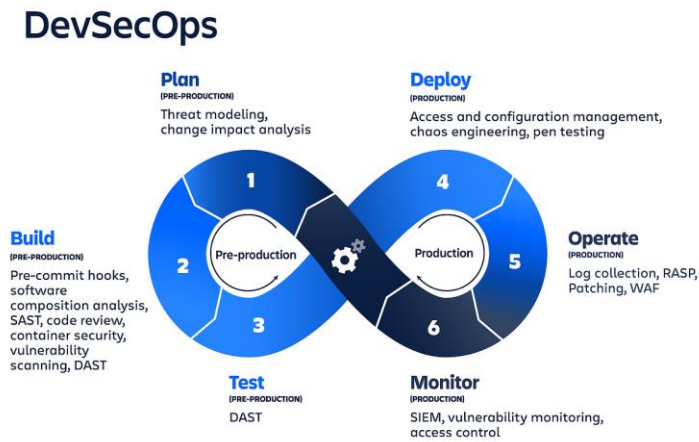


Figura 10: Metodología DevSecOps

Fuente: DevSecOps-Tools, Atlassian

2.3.6 Estándares de Verificación

En el desarrollo de software seguro, los estándares de verificación juegan un rol crucial al establecer lineamientos claros y prácticas recomendadas para garantizar la protección de las aplicaciones frente a amenazas. Dos estándares ampliamente reconocidos en el ámbito de la seguridad de aplicaciones son el Application Security Verification Standard (ASVS) y el Mobile Application Security Verification Standard (MASVS), ambos desarrollados por OWASP.

2.3.6.1 ASVS

El Estándar de Verificación de Seguridad de Aplicaciones (ASVS) desarrollado por OWASP proporciona una base para probar controles de seguridad técnica en aplicaciones web y define una lista de requisitos para guiar un desarrollo seguro de aplicaciones (OWASP, 2024).

OWASP detalla que los requisitos que se encuentran en el estándar fueron desarrollados con los siguientes objetivos en mente:

- **Utilizar como métrica:** proporcionar a los desarrolladores y propietarios de aplicaciones un criterio con el que evaluar el grado de confianza que se puede depositar en sus aplicaciones web.
- **Usar como guía:** proporcionar orientación a los desarrolladores de controles de seguridad sobre qué incorporar en los controles de seguridad para satisfacer los requisitos de seguridad de la aplicación.
- **Uso durante la adquisición:** proporciona una base para especificar los requisitos de verificación de seguridad de la aplicación en los contratos.

Este estándar proporciona pautas para verificar los siguientes aspectos de seguridad en las aplicaciones web:

1. Arquitectura, Diseño y Modelado de Amenazas
2. Autenticación
3. Gestión de Sesiones
4. Control de Acceso
5. Validación y Saneamiento
6. Criptografía Almacenada

7. Manejo de Errores y Registro
8. Protección de Datos
9. Comunicación
10. Código Malicioso
11. Lógica de Negocio
12. Archivos y Recursos
13. API y Servicios Web
14. Configuración

2.3.6.2 MASVS

El Estándar de Verificación de Seguridad para Aplicaciones Móviles (MASVS) es un marco integral creado por OWASP que establece lineamientos claros y prácticas recomendadas para analizar y fortalecer la seguridad de las aplicaciones móviles.

Este estándar se concibe como una herramienta fundamental que actúa como métrica, guía y punto de referencia para verificar la seguridad de estas aplicaciones, convirtiéndose en un recurso esencial para desarrolladores, propietarios de aplicaciones y expertos en seguridad (Schleier, Holguera, Mueller, Willemsen, & Beckers, 2024).

CAPÍTULO 3: PROPUESTA DE SOLUCIÓN

En el capítulo 3, "Propuesta de solución", se abordará la propuesta para resolver el problema identificado en el caso de estudio de Not Waste, relacionado con la falta de claridad sobre cómo defenderse de amenazas cibernéticas. Not Waste enfrenta incertidumbre sobre qué medidas de seguridad implementar y cómo hacerlo, debido a la ausencia de un plan de seguridad claro y a la dificultad de elegir un marco de seguridad adecuado, dada la amplia variedad disponible. Por ello, se propone el uso de SAMM como marco de seguridad, ya que se ajusta al contexto organizacional de Not Waste.

Este capítulo también ofrecerá una explicación detallada del marco de seguridad SAMM, su estructura y una visión simplificada del modelo, para resolver las preguntas planteadas en la problemática. Además, se describirá cómo implementar SAMM en las organizaciones.

Asimismo, se abordarán las fases previas a la implementación del modelo SAMM en Not Waste, que incluyen la planificación, evaluación y definición del plan y objetivos. Estas fases servirán de base para el capítulo 4, donde se continuará con el proceso de implementación de las prácticas.

3.1 Propuesta de solución

La problemática principal planteada en el texto es la dificultad de las organizaciones para protegerse de amenazas cibernéticas, debido a la falta de claridad sobre qué activos requieren protección, cómo deben protegerse, qué herramientas utilizar y cómo evaluar la efectividad de las medidas implementadas. Esta incertidumbre pone de manifiesto la necesidad de contar con un plan, estrategia y metodología claros que orienten la gestión de la ciberseguridad. Por ello, se propone el uso de SAMM como marco de seguridad para proporcionar una guía estructurada y efectiva.

OWASP SAMM (Software Assurance Maturity Model) es una metodología diseñada para ayudar a las organizaciones a integrar prácticas de seguridad en el ciclo de vida del desarrollo de software (SDLC). Proporciona un marco estructurado que permite evaluar el nivel actual de madurez en seguridad, identificar áreas de mejora y definir un plan de acción para implementar prácticas de seguridad de manera gradual.

3.2 Justificación de la Metodología

Se escogió OWASP SAMM como metodología ya que se destaca por su enfoque progresivo y adaptable, lo que la hace ideal para organizaciones en diferentes niveles de madurez. En lugar de imponer desde el inicio conceptos complejos o difíciles de aplicar, SAMM propone una implementación gradual y estructurada, permitiendo a las organizaciones comenzar

con pasos simples y avanzar progresivamente hacia niveles óptimos de madurez en seguridad. En la tabla 1, se puede ver una comparativa entre las principales opciones para escoger un marco de seguridad informática para aplicar en una startup tecnológica.

Tabla 1: Comparación de marcos de seguridad informática
Fuente: Elaboración propia

Característica	BSIMM	SAMM (OWASP)	ISO 27001
Enfoque	Medición de la madurez en la seguridad del software basado en prácticas observadas.	Modelo para implementar y mejorar la seguridad en el ciclo de vida del software.	Gestión de seguridad de la información mediante un Sistema de Gestión de Seguridad de la Información (SGSI).
Ámbito principal	Seguridad del software.	Seguridad del software.	Seguridad de la información a nivel organizacional.
Estructura	Basado en 12 prácticas organizadas en 4 dominios: Gobierno, Inteligencia, Defensas y SDLC seguro.	Basado en 5 dominios: Gobierno, Diseño, Implementación, Verificación y Operaciones.	Basado en controles y requisitos definidos en el Anexo A. Divido en 4 segmentos: Personas, Tecnológico, Organización, Físico.
Objetivo	Evaluar el estado actual de las prácticas de seguridad de software.	Proveer un marco para evaluar y mejorar la seguridad en el desarrollo de software.	Implementar y certificar un SGSI que garantice la protección de la información.
Certificación	No aplica.	No aplica.	Certificación formal otorgada por organismos acreditados.
Nivel de detalle	Observa prácticas existentes, sin prescribir cómo implementarlas.	Proporciona un marco práctico y flexible con niveles de madurez para cada práctica.	Especifica controles y requisitos obligatorios para la certificación.
Orientación a la mejora	Progresivo: permite mejorar desde niveles iniciales hasta avanzados.	Progresivo: permite mejorar desde niveles iniciales hasta avanzados.	Requiere un ciclo de mejora continua (PDCA).
Flexibilidad	Enfocado exclusivamente en seguridad de software.	Altamente flexible y adaptado al ciclo de vida del desarrollo de software.	Más rígido debido a su naturaleza de estándar formal.

Costo	organizaciones contratan consultores especializados o servicios de evaluación proporcionados por Synopsys	Gratuito y de código abierto.	Costos asociados a certificación y mantenimiento.
Uso	Usado principalmente por grandes organizaciones de software.	Adoptado por comunidades de desarrollo de software y pequeñas organizaciones.	Utilizado ampliamente en sectores empresariales que requieren certificación formal.

En el contexto de una startup tecnológica, SAMM resulta ser la mejor opción, ya que proporciona directrices clave para proteger el software desde etapas tempranas, priorizando los aspectos críticos y adaptándose a los recursos y capacidades de este tipo de organizaciones, donde se necesita implementar una seguridad de aplicaciones fuerte.

El enfoque de SAMM facilita la integración de la seguridad como un componente esencial del ciclo de vida del desarrollo de software, sin sobrecargar a los equipos con prácticas avanzadas desde el comienzo. Otra opción para considerar es OWASP ASVS, que es un estándar importante para verificar las aplicaciones, sin embargo, este estándar no es un marco de trabajo, que presente una metodología de implementación, aunque resulta extremadamente útil y se emplea para cumplir ciertos objetivos de SAMM, su alcance se limita a controles en aplicaciones web, sin abarcar aspectos como operaciones o aplicaciones móviles. En contraste, SAMM ofrece una estructura más amplia con una metodología definida y basada en niveles, lo que lo convierte en una herramienta valiosa para fortalecer las iniciativas de seguridad en una startup tecnológica.

,3.3 OWASP SAMM

SAMM es un modelo prescriptivo, un marco abierto que es simple de usar, completamente definido y medible. Los detalles de la solución son lo suficientemente fáciles de seguir incluso para personal que no esté relacionado con la seguridad. Ayuda a las organizaciones a analizar sus prácticas actuales de seguridad en el software, construir un programa de seguridad en iteraciones definidas, mostrar mejoras progresivas en las prácticas seguras y definir y medir actividades relacionadas con la seguridad. La siguiente imagen muestra la estructura general del modelo SAMM donde se logran apreciar sus prácticas:

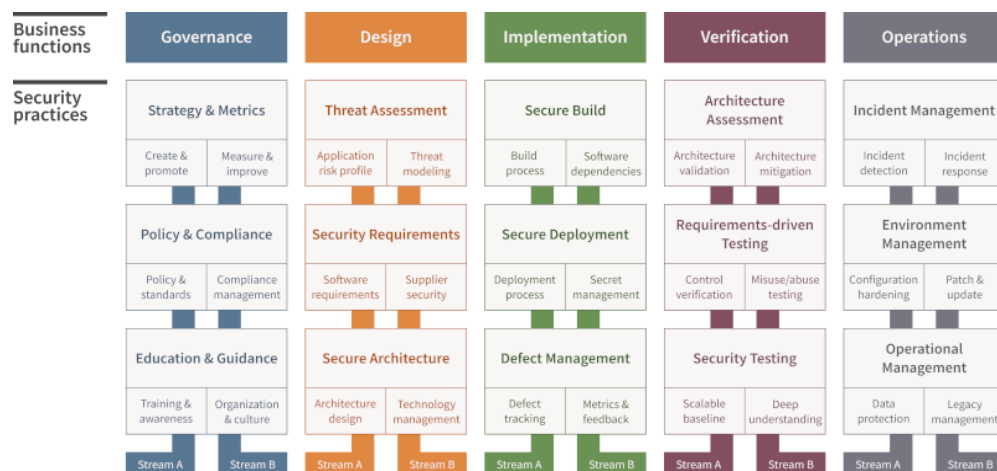


Figura 11: Modelo de madurez SAMM
Fuente: OWASP SAMM

El Software Assurance Maturity Model (SAMM) está basado en 15 prácticas de seguridad, agrupadas en 5 funciones empresariales clave. Cada práctica de seguridad contiene un conjunto de actividades, organizadas en 3 niveles de madurez. Las actividades de un nivel de madurez inferior suelen ser más fáciles de ejecutar y requieren menos formalización que las de los niveles superiores, la siguiente figura como está estructurada Cada una de las 5 funciones empresariales:

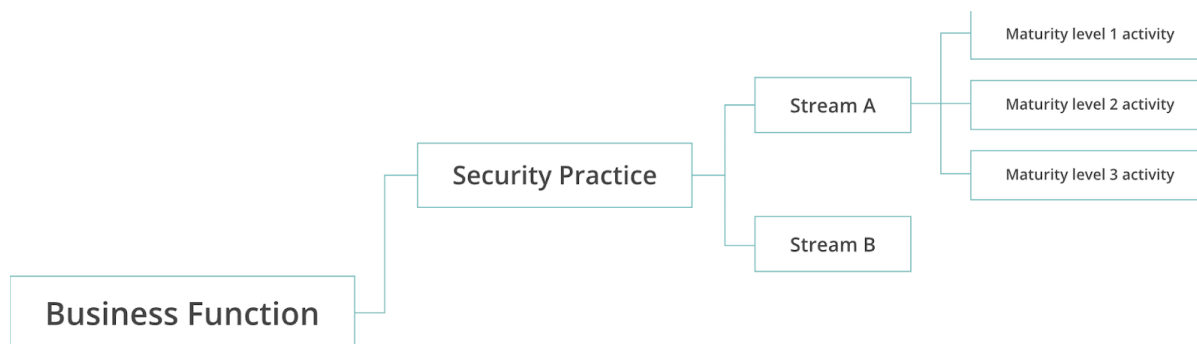


Figura 12: Estructura de SAMM
Fuente: OWASP SAMM

En la figura anterior se observa que cada práctica de seguridad está compuesta por dos flujos, los cuales abordan diferentes aspectos de esta y cuentan con objetivos específicos. Estos flujos se alinean y conectan las actividades a lo largo de los tres niveles de madurez, facilitando su implementación progresiva.

Criterios de calidad

En el modelo SAMM (Software Assurance Maturity Model), los criterios de calidad se utilizan para garantizar que las prácticas y controles implementados en el ciclo de vida del desarrollo de software cumplan con estándares definidos y sean efectivos para mitigar riesgos de seguridad. Su función principal es proporcionar una guía clara y objetiva para evaluar la calidad de las prácticas implementadas y verificar que se alineen con los objetivos de seguridad establecidos en cada etapa de madurez.

Funciones clave de los criterios de calidad en SAMM:

- **Medición del progreso:** Permiten evaluar si las prácticas están siendo adoptadas correctamente y si se están alcanzando los niveles de madurez definidos en el modelo.
- **Verificación:** Sirven como referencia para comprobar que los controles y procesos implementados son efectivos y cumplen con los requisitos de seguridad esperados.

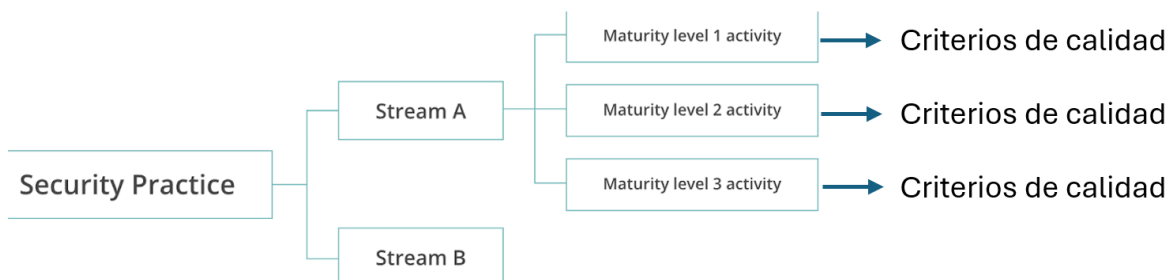


Figura 13: Criterios de calidad

Fuente: Adaptado de OWASP SAMM, 2024

Funciones empresariales y ciclo de vida del software

Las funciones empresariales definidas por SAMM (Gobernante, Diseño, Implementación, Verificación y Operaciones) están alineadas con las etapas del ciclo de vida del software, pero no son exactamente iguales. Mientras que el ciclo de vida del software se enfoca en el proceso técnico de desarrollar software (como planificación, diseño, desarrollo, pruebas e implementación).

3.4 Como implementar OWASP SAMM

Como explica OWASP SAMM (2024) en su guía de inicio rápido el proceso de implementación en una organización comienza con una fase inicial de preparación, donde se establecen las bases para su adopción. Posteriormente, se realiza una evaluación exhaustiva para identificar el estado actual de las prácticas de seguridad dentro del ciclo de vida del desarrollo de software. Con base en los resultados de esta evaluación, se definen metas claras y específicas que alineen las necesidades de la organización con los objetivos de seguridad.

Una vez establecidas las metas, se procede a planificar y ejecutar las actividades necesarias para alcanzar los objetivos planteados, incorporando mejoras en las prácticas existentes o implementando nuevas según sea necesario. Finalmente, se concreta la implementación de los cambios, asegurando su integración en los procesos organizacionales.

Lo que hace que OWASP SAMM sea especialmente eficaz es su enfoque en la mejora continua. Este marco permite repetir su ciclo de forma periódica, usualmente cada 3 a 12 meses, lo que garantiza una evolución constante de las prácticas de seguridad en respuesta a nuevos desafíos, tecnologías y necesidades organizativas.



Figura 14: Metodología de implementación de SAMM

Fuente: Adaptado de Guía de Inicio Rápido, por OWASP SAMM, 2024

A continuación, se presentarán los pasos previos a la implementación de las actividades técnicas establecidas por SAMM como prácticas de seguridad. Primero se explicará cómo realizar estos pasos, y posteriormente se detallará la implementación de los mismos, los cuales son: Preparar, Evaluar, Definir el plan y Establecer objetivos. En el capítulo 4 se abordarán las prácticas de seguridad que se establecieron como objetivos en estos pasos previos a la implementación.

3.4.1 Pasos de implementación

Los pasos de implementación describen en detalle las actividades clave que deben realizarse en cada una de las etapas definidas por la metodología OWASP SAMM, siguiendo las directrices de su guía de inicio rápido. La implementación está diseñada para proporcionar un marco estructurado que permita a las organizaciones integrar prácticas de seguridad en el ciclo de vida del desarrollo de software de manera eficiente y efectiva.

La descripción, que se presentara es explicada y adaptada de la información abierta que proporciona SAMM en “*Quick start guide for version 2.0*” que presenta una guía de cómo aplicar el modelo de madurez.

3.4.2 Preparación

El objetivo de esta etapa es garantizar un correcto inicio del proyecto, el éxito de la implementación de cualquier metodología depende en gran medida de un inicio bien estructurado. En el caso de la metodología OWASP SAMM, establecer una base sólida desde el principio es fundamental para asegurar que las actividades subsiguientes se realicen de manera alineada con los objetivos organizacionales y con un enfoque claro en la seguridad del software.

Las actividades clave de este proceso, son las siguientes:

- **Definir el alcance del proyecto:** El primer paso consiste en delimitar con precisión el alcance de la implementación. Esto implica determinar si la metodología se aplicará a nivel organizacional, a un equipo en particular, o a un proyecto específico.
- **Identificar a las partes interesadas:** Las partes incluyen tanto a los actores directamente involucrados en el desarrollo y la seguridad (como desarrolladores y especialistas en seguridad) como a los responsables de la toma de decisiones estratégicas (líderes de proyecto, gerentes y ejecutivos).
- **Socialización del proyecto:** La comunicación efectiva de los objetivos, beneficios y alcances del proyecto es fundamental para lograr la alineación y el apoyo de las partes interesadas.

3.4.3 Evaluación

Este paso consiste en identificar y comprender el nivel de madurez de las prácticas de seguridad de software dentro del alcance definido. Este proceso es fundamental para establecer un punto de partida que permita planificar mejoras estratégicas.

Las actividades clave de este proceso, son las siguientes:

- **Evaluar las prácticas actuales:** Esta actividad consiste en recopilar información sobre el estado actual de las prácticas de seguridad mediante entrevistas con partes interesadas clave, como desarrolladores y especialistas en seguridad. OWASP SAMM ofrece dos enfoques: uno detallado, basado en evidencia, y otro liviano, para una evaluación rápida.
- **Determinar el nivel de madurez:** Una vez evaluadas las prácticas, se determina el nivel de madurez utilizando el sistema de puntuación de SAMM. Las actividades se califican mediante opciones múltiples, y las puntuaciones se promedian por área para obtener una puntuación general, lo que ayuda a identificar áreas de mejora y priorizar esfuerzos.

3.4.3.1 Como realizar la evaluación

Para poder mejorar primero se debe conocer la situación actual en la que se encuentra la organización términos de seguridad, para ello se debe realizar una autoevaluación que permita obtener una descripción general del estado de las actividades de seguridad del software.

Para poder realizar esta evaluación OWASP SAMM ofrece una “caja de herramientas”, que son hojas de cálculo en dos versiones, Excel y Google Sheets, las cuales permiten medir la calidad y el desempeño de las actividades de seguridad de software en la organización, calculando el puntaje de madurez, al responder las preguntas contenidas en la sección de “Entrevista” o “Interview”. La siguiente figura presenta un diagrama que ilustra el contenido de la caja de herramientas y posteriormente se explica como realizar la evaluación respondiendo las preguntas de la entrevista.

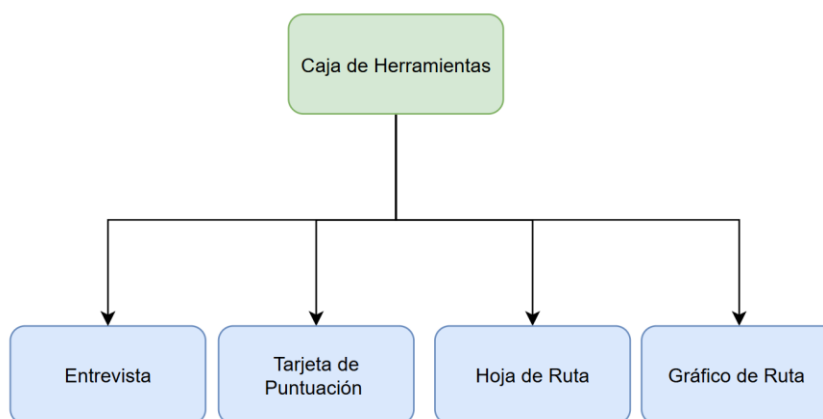


Figura 15: Contenido caja de herramientas, OWASP SAMM

Fuente: Elaboración propia

3.4.3.2 Entrevista

Entrevista es una pestaña de la planilla “caja de herramientas”, contiene una lista de preguntas diseñadas para evaluar el estado actual de la organización en términos de seguridad de aplicaciones. Cada pregunta está asociada a una actividad específica del Modelo OWASP SAMM. La figura siguiente presenta una representación de la estructura de entrevista (*interview*), cuyas secciones se encuentran directamente relacionadas con la estructura del modelo OWASPSAMM descrita en el capítulo anterior.

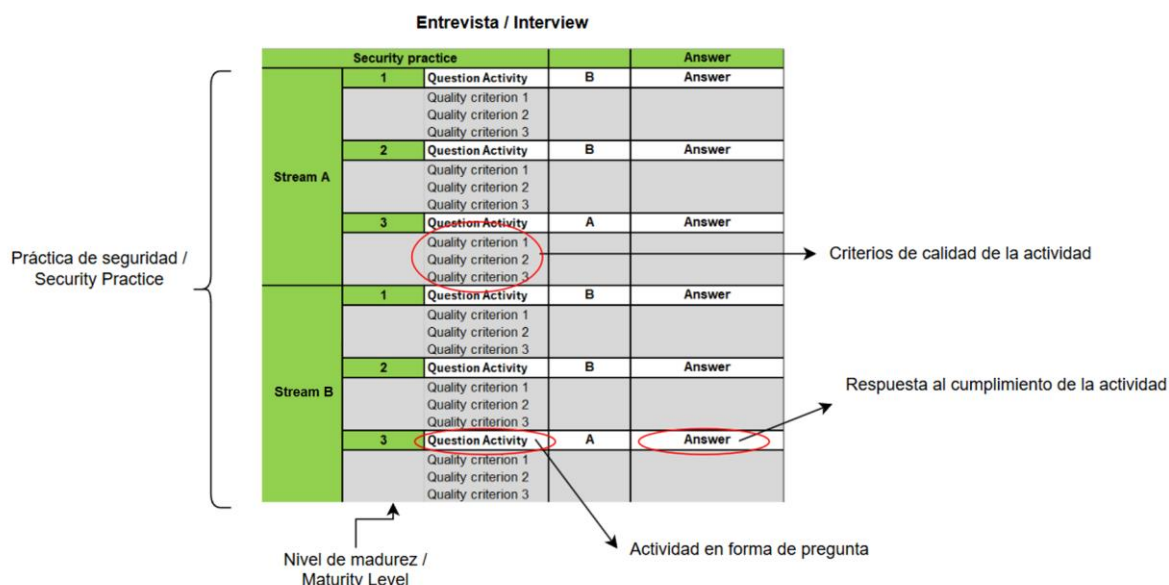


Figura 16: Estructura entrevista, autoevaluación SAMM

Fuente: Elaboración propia

La entrevista escribe las Maturity level activity⁴ (actividades), las cuales son las actividades más pequeñas escritas en “forma de pregunta”, no son exactamente la actividad en forma de pregunta, pero es similar, ejemplo:

- **Actividad:** Prueba de controles de seguridad del software.
- **Pregunta:** ¿Prueba las aplicaciones para comprobar el correcto funcionamiento de los controles de seguridad estándar?

⁴ Revisar estructura de OWASP SAMM explicadas anteriormente en la introducción de SAMM, para reconocer las *Maturity level Activities*, las cuales son las actividades más básicas del modelo.

Como responder: Para responder las preguntas de entrevista se deben considerar el cumplimiento de los criterios de calidad en la actividad, la siguiente tabla muestra cómo se debe responder y posteriormente un ejemplo.

Tabla 2: Como responder a las preguntas de SAMM.

Fuente: Elaboración propia.

Respuesta	Descripción
No	Se responde "No" si los criterios de calidad no se cumplen por completo
Otra Opción	Se responde "Otra Opción" si todos los criterios de calidad se cumplen plenamente.

Las opciones de respuesta pueden variar según la pregunta, pero a continuación se presentará un ejemplo basado en la actividad del flujo A (stream A) "Verificación de control" en el nivel de madurez 1. Primero, se mostrará la tabla 3 con la pregunta y los criterios de calidad, seguida de la tabla 4, en la que se explicará cómo responder a las preguntas.

Tabla 3: Ejemplo de pregunta y criterios de calidad.

Fuente: elaboración propia.

Pregunta (Actividad)	¿Prueba las aplicaciones para comprobar el correcto funcionamiento de los controles de seguridad estándar?
Criterios de calidad	- Las pruebas de seguridad al menos verifican la implementación de autenticación, control de acceso, validación de entrada, codificación y escape de datos y controles de cifrado. - Las pruebas de seguridad se ejecutan siempre que la aplicación cambia su uso de los controles.

En la tabla 4, se presentan las respuestas de la actividad perteneciente al flujo A (*stream A*) "Verificación de control" nivel de madurez 1:

Tabla 4: Ejemplo respuestas de actividad.

Fuente: Elaboración propia.

Respuesta	Descripción
No	Se responde "No" si los criterios de calidad no se cumplen por completo
Otras Opciones	- Sí, algunos de ellos - Sí, al menos la mitad de ellos - Sí, la mayoría o todos ellos

Las “otras opciones” no se refieren al cumplimiento de los criterios de calidad, si no al cumplimiento del total de la actividad al cumplir todos los criterios de calidad, referente a las aplicaciones que establecieron como parte del alcance de evaluación. Es importante recalcar que se responde “No”, si no se cumplieron todos los criterios de calidad de la actividad.

Al completar de responder el total de preguntas de la sección de entrevista en la “caja de herramientas” se debe pasar a ver la “tarjeta de puntuación” o “*Scorecard*”, para visualizar de forma gráfica el resultado de la evaluación.

3.4.3.3 Tarjeta de puntuación

La tarjeta de puntuación o “*Scorecard*” es una pestaña de la “caja de herramientas” la cual presenta datos importados automáticamente desde las hojas "Entrevista" y "Hoja de Ruta". Permite visualizar el nivel de madurez en la organización de forma visual, identificando el nivel de cumplimiento de las prácticas de seguridad. Se actualiza de manera automática cuando se realizan cambios en dichas hojas.

En esta pestaña de la “caja de herramientas”, se muestra el puntaje de madurez actual, así como gráficos relacionados. A continuación, se presenta una figura que ilustra cómo se muestran el puntaje y los gráficos en la hoja de cálculo:

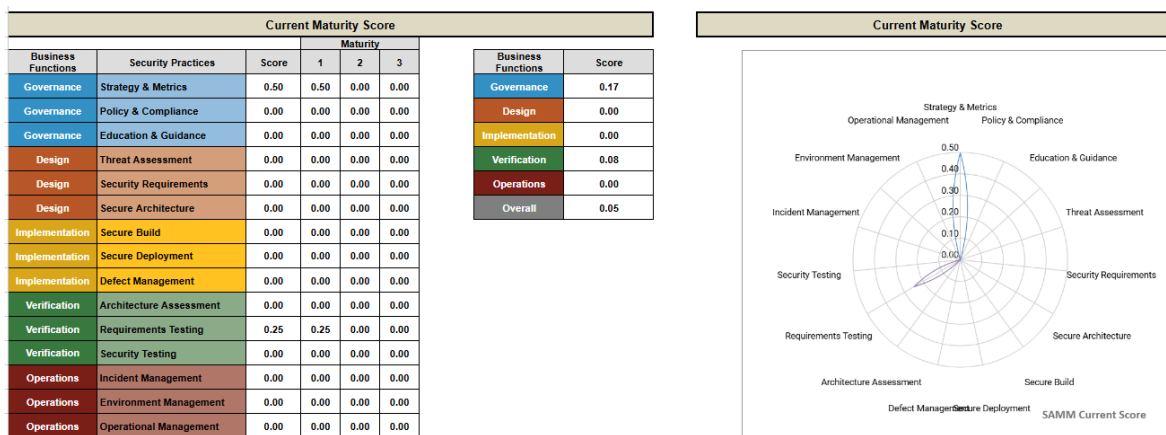


Figura 17: Tarjeta de puntuación "Scorecard", OWASP SAMM

Fuente: Elaboración propia

La tarjeta de puntuación refleja las calificaciones del nivel de madurez de la organización, calculadas a partir de las respuestas obtenidas en la entrevista. En la tabla ubicada a la izquierda de la figura X se detallan las puntuaciones por nivel de madurez, junto con un promedio del cumplimiento de las actividades según cada función de negocio (*business function*). Por su parte, el gráfico de la derecha visualiza las fortalezas y debilidades en el cumplimiento del modelo de madurez.

3.4.4 Establecer el objetivo

El presente proceso tiene como objetivo desarrollar una puntuación objetiva que sirva como criterio para guiar las decisiones y acciones dentro de la organización. Establecer una puntuación de referencia para identificar y priorizar las actividades más relevantes y alineadas con los objetivos estratégicos de la organización. Sirve para poder decidir cómo nos veremos en los siguientes meses, poniendo énfasis en las prácticas que queremos mejorar y que se establecieron como objetivos.

Sus actividades principales son:

- **Definir el objetivo:** El primer paso es establecer o actualizar el objetivo, identificando las actividades que deben implementarse en la organización. Generalmente, esto incluirá más actividades de nivel inferior (nivel de madurez) que de nivel superior. Es importante asegurarse de que las actividades seleccionadas tengan coherencia y considerar las dependencias entre ellas.
- **Estimar el impacto general:** Una vez definido el objetivo, se debe estimar el impacto de su implementación en la organización, tratando de expresarlo en términos presupuestarios. Esto ayudará a comprender los recursos necesarios y el valor que se generará a través de la ejecución de las actividades seleccionadas.

Para establecer el objetivo, OWASP SAMM recomienda utilizar el SAMM Benchmark, que proporciona datos de referencia de otras organizaciones en cuanto al cumplimiento de las prácticas de SAMM, sirviendo como una fuente comparativa. Además, se sugiere utilizar la hoja de ruta (*roadmap*), que forma parte de los recursos disponibles en la "caja de herramientas", para "calcular las mejoras en la puntuación de madurez en función de las respuestas futuras".

SAMM Benchmark es un conjunto de datos de referencia de las evaluaciones de otras organizaciones en cuanto al cumplimiento de las prácticas de seguridad que propone SAMM. El objetivo principal es permitir que las empresas se comparen con otras dentro de la industria. La siguiente figura muestra un resumen de los resultados de conjuntos de datos de referencia, proporcionado por SAMM:



Figura 18: Resultado generales SAMM Benchmark.
Fuente: Elaboración propia.

3.4.5 Definir el plan

El propósito de la definición del plan es desarrollar o actualizar el plan para llevar a la organización al siguiente nivel. Luego de haber definido hacia donde se quiere llegar, cual es el objetivo, se crea un plan para establecer como llegar hacia esos objetivos, cuáles serán las tareas que permitirán a la organización llegar a sus ideales propuestos, como hacer para poder completar los criterios de calidad para poder establecer una mejora, propuesta en la definición de los objetivos. Las siguientes actividades son fundamentales para establecer el plan de cumplimiento en esta etapa de implementación de SAMM:

- **Determinar el cronograma de cambios:** Se debe elegir una estrategia de cambio realista, considerando el número y la duración de las fases para realizar los cambios.
- **Desarrollar/actualizar el plan de la hoja de ruta:** Las actividades deben distribuirse a lo largo de las fases de la hoja de ruta, teniendo en cuenta el esfuerzo necesario para su implementación. Es importante equilibrar la carga de trabajo entre las fases y considerar las dependencias entre las actividades para una ejecución eficiente.

3.5 Planificación de implementación en Not Waste

La metodología propuesta por SAMM para aplicar el modelo de madurez, que incluye las etapas de preparación, evaluación frente al modelo, establecimiento de objetivos y planificación de la implementación, sirve como guía para la aplicación de prácticas de seguridad en el marco del modelo. A continuación, se detallará cómo se aplicaron cada uno de estos pasos para implementar dichas prácticas en *Not Waste*.

Las fases empleadas en *Not Waste* fueron:

1. Preparación y análisis del contexto
2. Evaluación frente al modelo
3. Establecimiento de objetivos
4. Planificación de la implementación
5. Implementación

3.5.1 Preparación y análisis de contexto

Este es primer paso para iniciar la implementación de las prácticas de seguridad de propuestas en SAMM, el objetivo de este paso es garantizar un correcto inicio del proyecto. En esta etapa se definirá el alcance de la implementación de seguridad y las partes interesadas del proyecto Not Waste.

Definición del alcance: El proyecto de Not Waste como se presentó en el capítulo 2.1 “Introducción al caso de estudio”, está compuesta por dos software creados por la organización, una aplicación móvil escrita en Flutter y una API que permite la interacción con datos y funciones necesarias para la lógica de negocio de la aplicación, el alcance de la implementación de prácticas de seguridad **solo corresponde a la plataforma API**.

Detalle API de Not Waste: La API (*Application Programming Interface*) está escrita en Django y se encuentra desplegado en un servidor Linux de AWS EC2 tipo de instancia t2.micro, utiliza Nginx como servidor web y utiliza Gunicorn un servidor WSGI (Web Server Gateway Interface) que actúa como intermediario entre el servidor web y la API Django⁵.

Identificar a las partes interesadas: Los posibles inversionistas, usuarios de la aplicación y el equipo de desarrollo son las partes interesadas clave para Not Waste. Actualmente, la aplicación no cuenta con inversionistas ni usuarios activos, pero el perfil de cliente más común al que está dirigida, es decir, los dueños y dueñas de casa, constituye uno de los principales grupos de interés. A continuación, se presenta una tabla con el equipo de Not Waste que representa de manera más concreta las partes interesadas.

Tabla 5: Equipo de Not Waste

Fuente: Elaboración propia

Nombre	Rol
Jehoshua Mendoza	Desarrollador Frontend
Jeremy Ramos	Ingeniero de Datos
Fabian Bustos	Ingeniero IT / Seguridad
Benjamín Aguilera	Diseñador UI/UX
Martin Fonseca	Desarrollador Backend

⁵ Para facilitar la comprensión del lector, en adelante se denominará API Django a la interfaz desarrollada por la organización utilizando el framework Django.

3.5.2 Evaluación frente al modelo

La evaluación frente al modelo consiste en identificar y comprender el nivel de madurez de las prácticas de seguridad de software dentro del alcance definido anteriormente en la etapa de planificación, es decir la API Django de Not Waste. Esta evaluación se realizó respondiendo las preguntas de la entrevista (*interview*) que proporciona SAMM en su “caja de herramientas”, la cual proporciona una puntuación en función de las prácticas de seguridad aplicadas. En el capítulo 3 propuesta de solución se abarca el entendimiento del método de aplicación de SAMM y se explica de manera detalla cómo responder a las preguntas de la “*interview*”.

Evaluación de prácticas actuales: De acuerdo con las actividades de seguridad propuestas por SAMM, no se ha implementado ninguna práctica hasta el momento. Por ello, todas las respuestas a las preguntas fueron registradas como "No" en cada actividad, resultando en una puntuación de 0. A continuación, se presenta una imagen que ilustra una parte de la entrevista y cómo las prácticas evaluadas obtuvieron una puntuación total de 0.

Governance									
Stream	Level	Strategy & Metrics	Answer					Interview Notes	Rating
Create and Promote	1	Do you understand the enterprise-wide risk appetite for your applications? You capture the risk appetite of your organization's executive leadership The organization's leadership vet and approve the set of risks You identify the main business and technical threats to your assets and data You document risks and store them in an accessible location	No	-	0	0.000			0.00
	2	Do you have a strategic plan for application security and use it to make decisions? The plan reflects the organization's business priorities and risk appetite The plan includes measurable milestones and a budget The plan is consistent with the organization's business drivers and risks The plan lays out a roadmap for strategic and tactical initiatives You have buy-in from stakeholders, including development teams	No	-	0	0.000			
	3	Do you regularly review and update the Strategic Plan for Application Security? You review and update the plan in response to significant changes in the business environment, the organization, or its risk appetite Plan update steps include reviewing the plan with all the stakeholders and updating the business drivers and strategies You adjust the plan and roadmap based on lessons learned from completed roadmap activities You publish progress information on roadmap activities, making sure they are available to all stakeholders	No	-	0	0.000			
Measure and Improve	1	Do you use a set of metrics to measure the effectiveness and efficiency of the application security program across applications? You document each metric, including a description of the sources, measurement coverage, and guidance on how to use it to explain application security trends Metrics include measures of efforts, results, and the environment measurement categories Most of the metrics are frequently measured, easy or inexpensive to gather, and expressed as a cardinal number or a percentage Application security and development teams publish metrics	No	-	0	0.000			0.00
	2	Did you define Key Performance Indicators (KPI) from available application security metrics? You defined KPIs after gathering enough information to establish realistic objectives You developed KPIs with the buy-in from the leadership and teams responsible for application security KPIs are available to the application teams and include acceptability thresholds and guidance in case teams need to take action Success of the application security program is clearly visible based on defined KPIs	No	-	0	0.000			
	3	Do you update the Application Security strategy and roadmap based on application security metrics and KPIs? You review KPIs at least yearly for their efficiency and effectiveness KPIs and application security metrics trigger most of the changes to the application security strategy	No	-	0	0.000			
Policy & Compliance									
	1	Do you have and apply a common set of policies and standards throughout your organization? You have adopted existing standards appropriate for the organization's industry to account for domain-specific considerations Your standards are aligned with your policies and incorporate technology-specific implementation guidance	No	-	0	0.000			0.00

Figura 19: Evaluación de madurez en seguridad API (Not Waste)

Fuente: Elaboración propia

Determinar el nivel de madurez: El nivel de madurez de seguridad en cuanto al proyecto de Not Waste, es nulo. La siguiente figura muestra la puntuación global obtenida en base a la evaluación de las prácticas.

Business Functions	Security Practices	Score	Maturity		
			1	2	3
Governance	Strategy & Metrics	0.00	0.00	0.00	0.00
Governance	Policy & Compliance	0.00	0.00	0.00	0.00
Governance	Education & Guidance	0.00	0.00	0.00	0.00
Design	Threat Assessment	0.00	0.00	0.00	0.00
Design	Security Requirements	0.00	0.00	0.00	0.00
Design	Secure Architecture	0.00	0.00	0.00	0.00
Implementation	Secure Build	0.00	0.00	0.00	0.00
Implementation	Secure Deployment	0.00	0.00	0.00	0.00
Implementation	Defect Management	0.00	0.00	0.00	0.00
Verification	Architecture Assessment	0.00	0.00	0.00	0.00
Verification	Requirements Testing	0.00	0.00	0.00	0.00
Verification	Security Testing	0.00	0.00	0.00	0.00
Operations	Incident Management	0.00	0.00	0.00	0.00
Operations	Environment Management	0.00	0.00	0.00	0.00
Operations	Operational Management	0.00	0.00	0.00	0.00

Business Functions	Score
Governance	0.00
Design	0.00
Implementation	0.00
Verification	0.00
Operations	0.00
Overall	0.00

Figura 20: Puntuación de evaluación inicial de Not Waste

Fuente: Elaboración propia

3.5.3 Establecer objetivos

En esta etapa se definirán los objetivos que guiarán a la organización Not Waste en la implementación de las actividades necesarias. Estos objetivos permitirán identificar y priorizar las acciones más relevantes para la organización. En esencia, los objetivos representan las actividades que se busca implementar.

Definición de objetivos: Los objetivos que se desea establecer para implementar son los siguientes:

- Establecer patrones de diseño comunes y soluciones de seguridad para su adopción.
- Prueba de controles de seguridad del software.

Los objetivos se detallarán en la tabla 3, donde se identificarán las funciones de negocio a las que pertenecen y se asignará un código a cada uno para facilitar su identificación durante la implementación. A continuación, se explica el formato de los códigos utilizados en la tabla:

- El primer código, SA-AD, indica que la actividad pertenece a la práctica de seguridad *Secure Architecture* (SA), dentro del flujo *Architecture Design* (AD), y corresponde al nivel de madurez 2.
- El segundo código, RT-CV, señala que la actividad pertenece a la práctica de seguridad *Requirements-driven Testing* (RT), en el flujo *Control Verification* (CV), y corresponde al nivel de madurez 1.

Tabla 6: Objetivos de implementación

Fuente: Elaboración propia

Proceso	Código	Madurez	Objetivos
Diseño	SA-AD	2	Establecer patrones de diseño comunes y soluciones de seguridad para su adopción.
Verificación	RT-CV	1	Prueba de controles de seguridad del software.

3.5.5 Planificar la implementación

El objetivo de este paso es desarrollar el plan para mejorar Not Waste en cumplimiento de prácticas de seguridad, se determinaron cuáles eran las tareas que permitían a la organización llegar a los objetivos propuestos, esta sección es representativa al contexto de Not Waste considerando sus recursos.

Tabla 7: Plan de actividades SAMM.

Fuente: Elaboración propia.

Función de negocio	Código	Actividades	Duración (Días)
Diseño	SA-AD-2		
	SA-AD-2.1	Identificar servicios de seguridad reutilizables	13
	SA-AD-2.2	Categorizar los recursos según el mecanismo de seguridad que cumplen	4
	SA-AD-2.3	Evaluar la utilidad de cada recurso desde la perspectiva de los equipos de producto	4
	SA-AD-2.4	Seleccionar y estandarizar servicios compartidos	4
	SA-AD-2.6	Implementar soluciones de componentes reales	20
Verificación	RT-CV-1		45
	RT-CV-1.1	Identificar y evalúa referencias que ayuden a realizar las pruebas	7
	RT-CV-1.2	Identificar y seleccionar los controles necesarios	13
	RT-CV-1.3	Se escriben casos de prueba para los diferentes controles	13
	RT-CV-1.4	Realizar pruebas de autenticación.	4
	RT-CV-1.5	Realiza pruebas controles de acceso.	3
	RT-CV-1.6	Realiza pruebas de entrada de datos.	3
	RT-CV-1.7	Realiza prueba de codificación y escape de datos.	3
	RT-CV-1.8	Realizar pruebas de los controles de cifrado	3
			50
		Total	94

A continuación, se presentará el cronograma realizado utilizando SmartSheet, que ofrece un software similar a Excel con plantillas diseñadas para gestión de proyectos como los cronogramas, es muy útil para hacer cronogramas de manera rápida, sin necesidad de configuraciones como se haría en Excel, pero su capa gratuita solo cuenta con 30 días de prueba y para tener el plan pro son 9 dólares por mes:

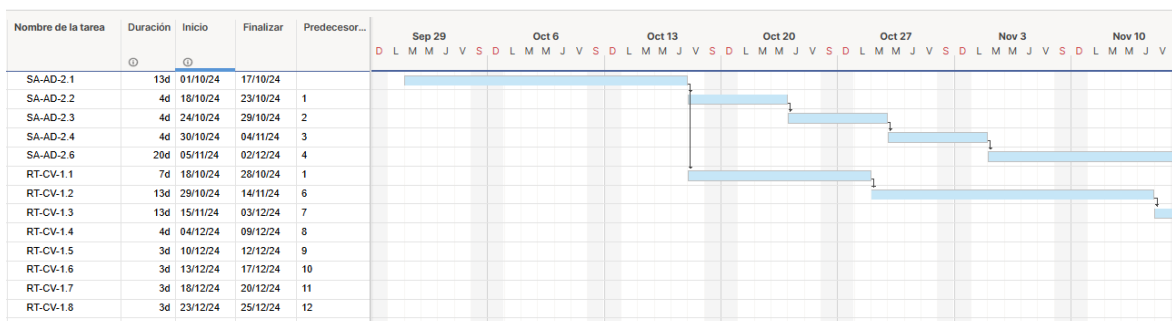


Figura 21: Cronograma de actividades SAMM.

Fuente: Elaboración propia.

La duración de las actividades depende directamente de la cantidad de personas involucradas en su ejecución. Las estimaciones de tiempo fueron propuestas antes de implementar la solución, por lo que es posible que, en ausencia de personal capacitado para llevar a cabo las tareas, el tiempo requerido sea diferente al previsto, pudiendo extenderse o reducirse según las circunstancias.

Este cronograma no cuenta con una estrategia de creación, como las que se aplican en una gestión de proyectos formal. Sin embargo, proporciona una perspectiva global de las actividades. A medida que se comprendan mejor los conceptos y objetivos propuestos, es probable que el cronograma sufra ajustes debido a posibles subestimaciones iniciales, especialmente en el caso de organizaciones pequeñas, como es el caso de Not Waste.

CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN

4.1 Introducción al capítulo

En este capítulo se analizará la implementación de las prácticas definidas en los objetivos del marco de seguridad de aplicaciones, específicamente el diseño de arquitectura y la verificación de requisitos. Primero, se presentará el contexto previo a la implementación y luego se detallará cómo se aplicaron en Not Waste.

4.2 Práctica SA-AD-2

Como se define en el modelo de madurez SAMM, la práctica de *Secure Architecture* (SA) se centra en la seguridad vinculada a los componentes y la tecnología con los que trabaja durante el diseño arquitectónico de su software. El objetivo del flujo *Architecture Design*, correspondiente al nivel de madurez 2 (AD-2), es establecer patrones de diseño comunes y soluciones de seguridad que faciliten su adopción.

Explicación de actividad: Como lo explica el marco, lo que debe realizarse en esta práctica es identificar infraestructura o los servicios compartidos con funcionalidad de seguridad. Como por ejemplo servicios de inicio de sesión único (SSO), servicios de control de acceso o de derechos, servicios de registro y monitoreo o firewall a nivel de aplicación.

Criterios de calidad: Los requisitos de calidad que deben cumplirse para esta actividad, son:

- Tiene una lista documentada de servicios de seguridad reutilizables, disponible para las partes interesadas relevantes
- Ha revisado la postura de seguridad de referencia para cada servicio seleccionado
- Sus diseñadores están capacitados para integrar cada servicio seleccionado siguiendo las pautas disponibles.

Al final de la explicación de esta práctica se evaluará el cumplimiento de los criterios de calidad.

El proceso seguido para realizar la actividad consistió inicialmente en identificar los servicios de seguridad, categorizarlos y seleccionar aquellos que se consideraron más adecuados, basándose en ciertos criterios que se explicarán más adelante. Finalmente, se procedió a implementar los servicios seleccionados, incorporando las funcionalidades de seguridad elegidas.

4.2.1 Identificación de servicios de seguridad

En la identificación de servicios de seguridad de la práctica de seguridad SA-AD-2, se han identificado distintos servicios con funcionalidades de seguridad, entre las que se incluyen

la autenticación, autorización, monitoreo y registros, firewall de aplicaciones (WAF), sistemas de detección de intrusiones (IDS) y sistemas de prevención de intrusiones (IPS). La identificación y análisis es crucial para contextualizar las alternativas de diseño y servicios de seguridad más comunes, así como para entender los beneficios de su implementación.

Antes de la identificación de servicios se abordarán conceptos básicos que faciliten la comprensión de las funcionalidades de seguridad utilizadas. Los conceptos están agrupados en términos de autenticación, firewall de aplicaciones, monitoreo y registro. En la siguiente figura, se presentará un diagrama que muestra los conceptos que serán definidos:

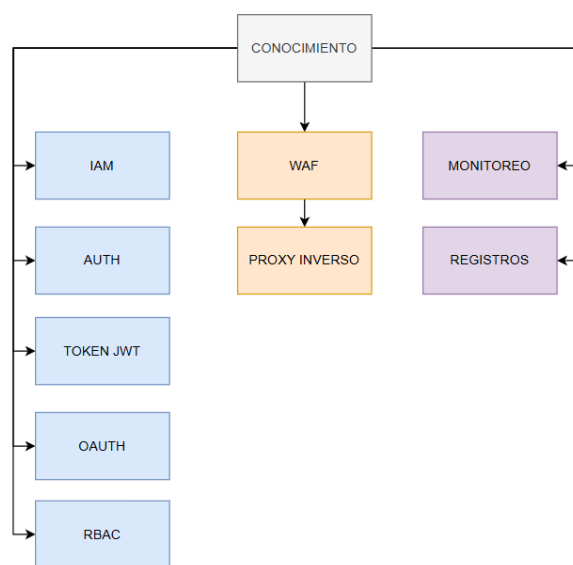


Figura 22: Conceptos previos a la identificación de servicios de seguridad.

Fuente: Elaboración propia.

Autenticación y autorización

En primer lugar, se debe tener conocimiento de los conceptos básicos de autenticación y autorización, para la aplicación de funciones de seguridad relacionadas ellas, como JWT, OAuth, RBAC o IAM. El artículo “autenticación frente a autorización” (Auth0, 2024) explica los conceptos de autenticación y autorización de la siguiente manera:

- **Autenticación:** Es el proceso de confirmación de la identidad de un usuario o un dispositivo. Durante el proceso de autenticación, una entidad suele basarse en alguna prueba para verificar quien es, es decir, un factor de autenticación. Por ejemplo, el carnet de identidad en Chile afirma de cierta manera que quienes somos.

- **Autorización:** La autorización por otro lado se refiere al proceso de verificar a qué recursos pueden acceder las entidades (usuarios o dispositivos), o qué acciones pueden realizar, es decir, sus derechos de acceso.

Ambos conceptos están relacionados con la seguridad, ya que tal como se menciona por parte de Microsoft en *what is authentication*, la autenticación es una parte importante de la ciberseguridad porque “la mayor prioridad de un infiltrado es obtener acceso no autorizado a los sistemas”.

Otros conceptos esenciales y ligados con la autenticación y autorización son los siguientes:

IAM: IAM (Gestión de Identidad y Acceso) es una tecnología que permite controlar la autenticación de usuarios y la autorización para acceder a los recursos. A través de IAM, se asegura que las personas correctas puedan acceder a los recursos digitales correspondientes en el momento oportuno y por las razones adecuadas.

RBAC: El término Control de acceso basado en roles (RBAC) se refiere a una estrategia de autorización que organiza los privilegios en función de un rol (de ahí el prefijo "basado en roles").

OAuth 2.0: OAuth 2.0 es un protocolo de autorización que permite a un usuario otorgar acceso a sus recursos protegidos a una aplicación de terceros, sin revelar sus credenciales. En lugar de compartir las credenciales del usuario, se emite un token de acceso que define qué recursos se pueden acceder, por cuánto tiempo y en qué condiciones. Auth0, por ejemplo, utiliza este sistema para generar tokens de acceso en formato JWT (JSON Web Token) para autorizar el acceso a APIs. Los "alcances" en OAuth son los permisos específicos que el token de acceso permite, y son determinados por el usuario al autorizar la aplicación.

JWT: Un JWT (JSON Web Token) es un estándar para crear tokens de acceso definidos en el RFC 7519, permiten la transmisión segura de información entre dos partes, como un cliente y un servidor, por ejemplo, en el contexto de Not Waste, el cliente sería la app móvil y el servidor la API Django.

Estructura: La estructura de un JWT podría verse de la siguiente manera, por lo general está compuesto por tres partes el Header, Payload y Signature, el token viene codificado y al ser decodificado, se puede ver su estructura, como en la siguiente figura:

Encoded
PASTE A TOKEN HERE

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG91IiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36P0k6yJV_adQssw5c
```

Decoded
EDIT THE PAYLOAD AND SECRET

HEADER: ALGORITHM & TOKEN TYPE

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

PAYLOAD: DATA

```
{
  "sub": "1234567890",
  "name": "John Doe",
  "iat": 1516239022
}
```

VERIFY SIGNATURE

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  your-256-bit-secret
) ☐ secret base64 encoded
```

Figura 23: Estructura JWT.

Fuente: Elaboración propia.

Claims: es una afirmación de información sobre un sujeto como un usuario o entidad, por ejemplo, sub (subject): Google-oauth2 | 908034111254, esta afirmación identifica al usuario.

Claims Set: El Conjunto de Declaraciones (Claims Set) de un JWT representa un objeto JSON cuyos miembros son las declaraciones (Claims) transmitidas por el JWT, los Claims estándar son:

- iss (Issuer): Identifica quién emitió el token. Por ejemplo, el servidor o servicio que generó el JWT.
- sub (Subject): Define el sujeto del token, como el usuario o entidad al que se refiere.

- aud (Audience): Especifica para quién está destinado el token (por ejemplo, un cliente o aplicación).
- exp (Expiration Time): Indica cuándo expira el token. Se expresa como un timestamp UNIX.

Monitoreo y Registros

Los conceptos de monitoreo y los registros están estrechamente vinculados, y su implementación ofrece beneficios significativos, como la identificación de posibles vulnerabilidades. En adelante se describirán estos conceptos.

- **Monitoreo:** El diccionario de la lengua española especifica que monitorear es “Observar mediante aparatos especiales el curso de uno o varios parámetros fisiológicos o de otra naturaleza para detectar posibles anomalías”.
- **Monitoreo de infraestructura:** En términos informáticos está presente el monitoreo de infraestructura, el cual “implica la observación, identificación, categorización y análisis continuos de cambios significativos en el estado que afectan los componentes de infraestructura”. (Splunk, 2024)
- **Registros:** Registro o log es un registro de los eventos que ocurren dentro de los sistemas y redes de una organización. (NIST) Los registros están compuestos por entradas (log entries); cada entrada contiene información relacionada con un evento específico que ha ocurrido en un sistema o red. Un ejemplo de estos registros puede ser uno generado por un servidor web como Nginx, donde se detallen eventos de peticiones al servidor.

El monitoreo de la infraestructura es esencial para identificar incidentes de seguridad y asegurar el cumplimiento de las políticas establecidas. Al supervisar registros del sistema, tráfico de red y eventos relacionados con la seguridad, las organizaciones pueden detectar actividades inusuales, posibles vulnerabilidades o infracciones, y actuar de manera oportuna para mitigar riesgos. (IBM, n.d)

Firewall de aplicaciones

Proxy inverso: Un proxy inverso es un servidor que se sitúa delante de los servidores web y reenvía las solicitudes del cliente (por ejemplo, el navegador web) a esos servidores web. (Cloudflare, 2024) Cuando los clientes realizan solicitudes a sitios y servicios en Internet, el servidor proxy intercepta esas peticiones y luego se comunica con los servidores web en nombre de esos clientes, como un intermediario. Esto puede ser útil para agregar medidas de seguridad antes de que las peticiones lleguen al servidor de destino.

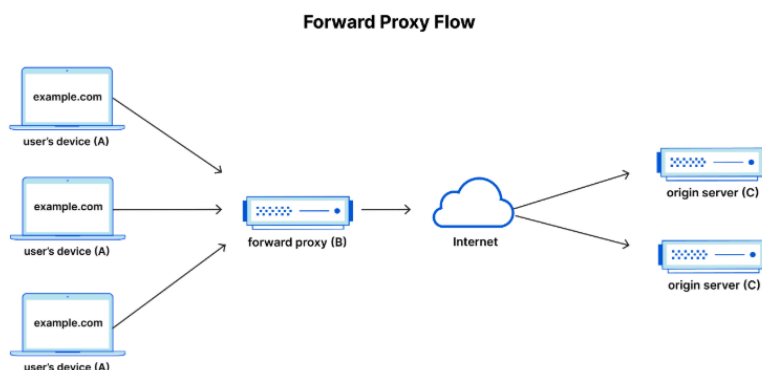


Figura 24: Proxy inverso.

Fuente: Cloudflare. (2024). ¿Qué es un Proxy inverso?

WAF: Cloudflare define WAF (Firewall de aplicaciones) en el artículo ¿Qué es un WAF? (Cloudflare, 2024) como una defensa del protocolo de capa 7 (en el modelo OSI). Al desplegar un WAF en una aplicación web, se coloca un “escudo” entre la aplicación web e Internet que ayuda a proteger las aplicaciones web al filtrar y monitorizar el tráfico HTTP proveniente de internet. La siguiente ilustración ofrece una representación visual de esta definición:

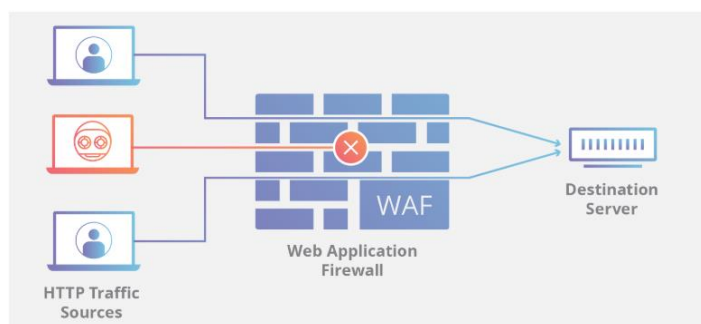


Figura 25: Firewall de aplicaciones.

Fuente: Elaboración propia.

IDS/IPS

Un sistema de detección de intrusiones (IDS) es una herramienta de seguridad de red que monitoriza el tráfico y los dispositivos de la red en busca de actividades maliciosas conocidas, actividades sospechosas o infracciones de las políticas de seguridad. (IBM)

Un sistema de prevención de intrusiones (IPS) supervisa el tráfico de red en busca de amenazas potenciales y las bloquea automáticamente mediante alertas al equipo de seguridad, la terminación de conexiones peligrosas, la eliminación de contenidos maliciosos o la activación de otros dispositivos de seguridad. (IBM)

4.2.2 Categorización de servicios

Como se muestra en la siguiente tabla se presentan algunos servicios con funciones de seguridad asociados agrupados por categorías de autenticación y autorización, monitoreo y auditoría, firewall de aplicaciones (WAF), e IDS/IPS. Luego de la tabla se introducirán los servicios seleccionados y su aporte.

Tabla 8: Categorización de servicios de seguridad.

Fuente: Elaboración propia.

Categoría	AWS	Alternativas	Funciones de seguridad
Autenticación y Autorización	Amazon Cognito	Auth0, Okta, Firebase Auth	Proveen inicio de sesión seguro y gestión de identidades.
Monitoreo y auditoría	AWS Cloudwatch	Splunk, Sentry, BetterStack, ELK Stack	Registran y monitorean eventos de seguridad.
WAF	AWS WAF	Cloudflare WAF	Protegen contra ataques a nivel de aplicación.
IDS/IPS	AWS GuardDuty	Snort, Suricata, Fail2Ban	Detectan y previenen intrusiones en redes o sistemas

4.2.3 Selección de servicios

Con base en los servicios y funcionalidades de seguridad previamente identificados, se procedió a seleccionar aquellos que resultaban más convenientes para el negocio, teniendo en cuenta el contexto de rápida implementación y limitaciones de recursos económicos.

Los criterios para la selección de estas herramientas incluyeron la disponibilidad de documentación, la facilidad de integración y la posibilidad de contar con una capa gratuita para pruebas de concepto. Cabe destacar que, en escenarios con mayores recursos y presupuestos, podrían considerarse herramientas con funcionalidades más avanzadas. No obstante, las herramientas seleccionadas son altamente recomendables dado que cumplen con los criterios establecidos.

En la Tabla 6 se presentan la selección de los servicios y las consideraciones clave que fueron tomadas en cuenta para la implementación del proyecto Not Waste.

Tabla 9: Selección de servicios de seguridad.

Fuente: Elaboración propia.

Categoría	Selección	Documentación	Integración	Capa gratuita
Autenticación y Autorización	Auth0	Documentación excelente y muy variada	Se integra perfectamente con Django, Flutter, Nginx, Cloudflare	25.000 usuarios activos al mes, inicio de sesión Google, Facebook
Monitoreo y auditoría	BetterStack	Documentación media	Integración simple con Django, Nginx y Linux	3 GB de registros ingeridos por mes retenidos durante 3 días
WAF	Cloudflare	Documentación excelente, comunidad de desarrolladores grande	Integración con todos los servidores	DDos protection, WAF, DNS, SSL

4.2.4 Detalle de servicios

Para facilitar una comprensión más detallada de los servicios utilizados, se presenta a continuación una descripción de cada uno. Este análisis será fundamental para la posterior implementación de sus funcionalidades de seguridad en la aplicación Not Waste, el siguiente diagrama muestra los servicios utilizados, lo cuales fueron mencionados previamente.

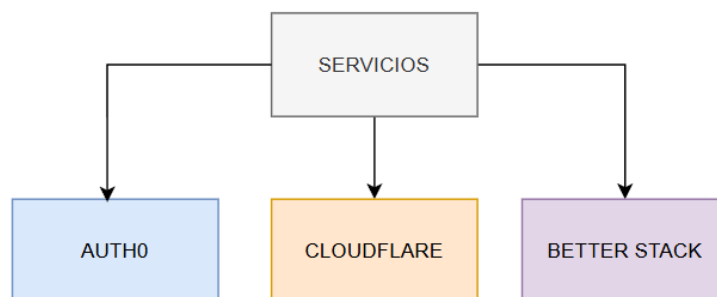


Figura 26: Diagrama general de servicios utilizados.

Fuente: Elaboración propia.

Auth0

El artículo “Introduction to Auth0” (Auth0, 2024) se describe el servicio de Auth0 que se define como un proveedor de gestión de acceso a identidades (IAM), el cual ayuda a gestionar cómo los usuarios acceden a las aplicaciones y servicios, protegiendo esos recursos y asegurando que solo los usuarios autorizados puedan acceder a ellos.

Auth0 es compatible con diversos tipos de aplicaciones y marcos de trabajo. Ya sea una aplicación web tradicional, una aplicación móvil o una aplicación de máquina a máquina, Auth0 proporciona configuraciones personalizadas para garantizar la implementación del flujo de autorización más seguro en cada caso.

Ejemplos de cómo Auth0 hace más seguro el flujo de autenticación:

- **Autenticación multifactor (MFA):** Auth0 permite agregar un segundo factor de autenticación, como códigos enviados por SMS, aplicaciones de autenticación (Google Authenticator) o notificaciones push. Esto asegura que incluso si un atacante obtiene la contraseña de un usuario, no podrá acceder a la cuenta sin el segundo factor.
- **Soporte para estándares modernos de seguridad:** Auth0 implementa protocolos de autenticación como OAuth 2.0, OpenID Connect y SAML. Estos estándares reducen

el riesgo de ataques como el phishing, ya que los tokens de acceso y autorización se manejan de manera segura.

- **Roles y permisos personalizados:** Auth0 permite asignar roles específicos a los usuarios, restringiendo el acceso a ciertas partes de la aplicación en función de sus permisos. Esto implementa un modelo de seguridad de privilegios mínimos, reduciendo la superficie de ataque.

Cloudflare: Cloudflare es un proveedor de servicios de infraestructura de seguridad web, ofrece productos para mejorar el rendimiento y la seguridad de un sitio web. Sus productos más conocidos son: Red de Entrega de Contenidos (CDN), Servicios de DNS, Firewall de aplicaciones web.

En el contexto de aplicación de Not Waste, el servicio de Cloudflare que se utilizará es el firewall de aplicaciones web (WAF) y la capa gratuita de su servicio de analítica, que permite visualizar las peticiones realizadas.

- **Reglas de Cloudflare:** el servicio ofrece una serie de reglas predefinidas para proteger tu aplicación web. Activa las reglas estándar de OWASP para protegerte contra amenazas comunes como inyecciones SQL, XSS, CSRF, entre otros.

Better Stack: (Better Stack, 2024) es una plataforma que combina herramientas para el monitoreo y análisis de sistemas con el objetivo de facilitar la gestión de incidentes y el análisis de registros en proyectos tecnológicos.

En Not Waste, el principal propósito de Better Stack será centralizar los registros generados por la API de Django. Su configuración y uso se explicarán en detalle en la sección dedicada a la implementación de monitoreo y registro.

Dos de sus principales productos son:

- **Uptime:** Una herramienta diseñada para garantizar la disponibilidad de servicios web mediante el monitoreo continuo de sitios y servidores. Ofrece alertas a través de múltiples canales y facilita la gestión de incidentes mediante rotaciones de guardia e integraciones con herramientas como Slack. La siguiente figura muestra como se ve el panel del servicio de Uptime.

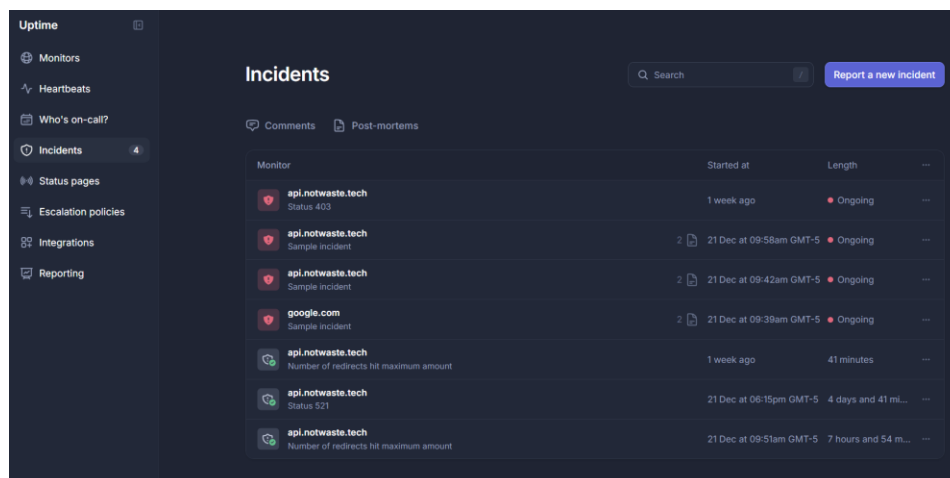


Figura 27: Uptime de Better Stack.

Fuente: Elaboración propia.

- Logtail/Telemetry:** Centraliza y estructura los registros del sistema, permitiendo su análisis mediante consultas SQL. Esto facilita la identificación de problemas y patrones, proporcionando una visión completa del sistema para mejorar la depuración, auditoría y optimización del rendimiento. La siguiente figura muestra cómo se ven los registros de manera centralizada en el servicio de Logtail/Telemetry.

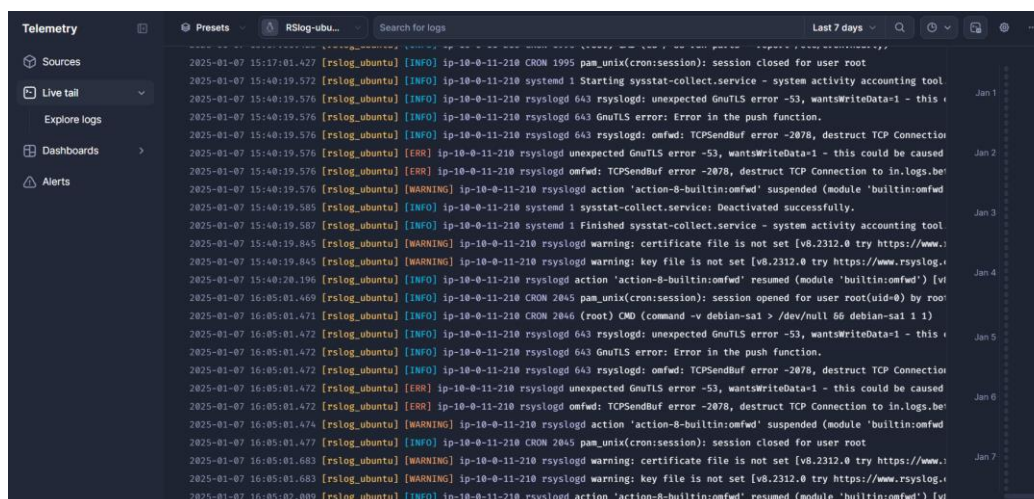


Figura 28: Logtail de Better Stack.

Fuente: Elaboración propia.

4.2.5 Implementación

Se detallará cómo se integraron los servicios seleccionados de Auth0, Cloudflare (WAF) y Better Stack con la infraestructura existente. La siguiente imagen ilustra las diferencias entre la infraestructura anterior y la actual, destacando las nuevas implementaciones realizadas en la práctica de seguridad correspondiente a SA-AD-2.

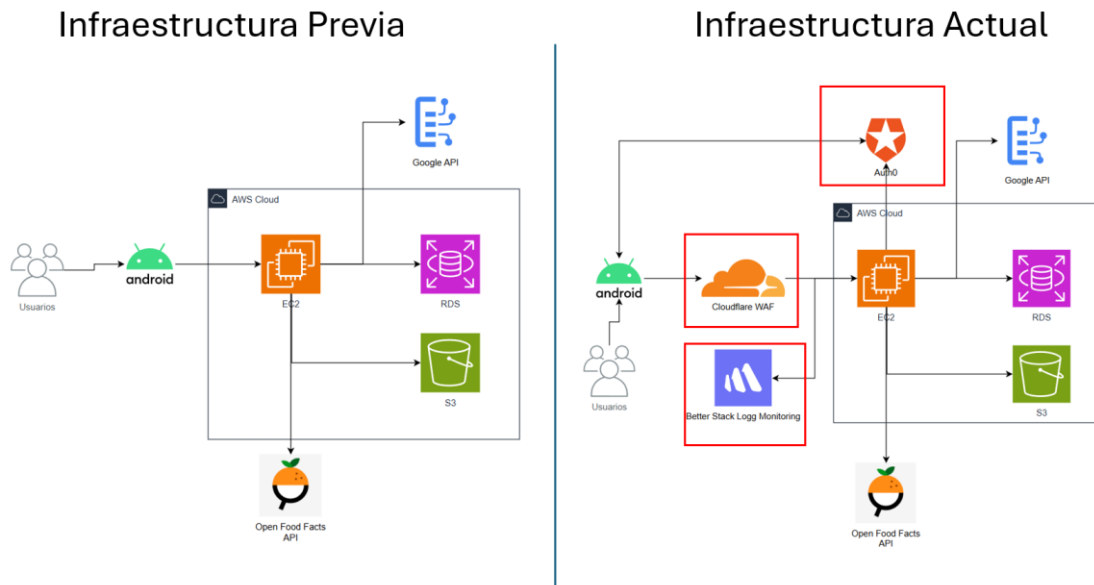


Figura 29: Comparación de infraestructura, previa y actual.

Fuente: Elaboración propia.

4.2.5.1 Autenticación y autorización

Para implementar funcionalidades de seguridad en el proyecto referentes a autenticación y autorización, se utiliza **Auth0**, aprovechando sus capacidades de control de acceso basado en roles (RBAC) y OAuth para una gestión segura y eficiente de los permisos y la autenticación.

Para explicar la implementación del servicio de autenticación y autorización Auth0, primero se explicará cual es el contexto anterior en el que se encontraba la comunicación entre el cliente y el servidor, luego se explicará el nuevo flujo de comunicación y posteriormente las configuraciones necesarias.

Contexto anterior al servicio Auth0

Antes de implementar Auth0, no se contaba con un mecanismo seguro para autenticar usuarios entre la aplicación móvil y la API. La autenticación se realizaba mediante un inicio de sesión básico, donde las credenciales se enviaban a través de una ruta específica y se almacenaban directamente en la base de datos.

El sistema se componía de un único servidor (API Django) encargado tanto de la autenticación de usuarios como de la lógica de negocio, incluyendo el acceso a recursos y datos. Este enfoque presentaba importantes riesgos de seguridad, ya que concentrar todas las funciones en un solo punto implicaba que cualquier vulnerabilidad podría comprometer todo el sistema. Esto incluía la exposición de las credenciales de los usuarios y de la información almacenada. Además, centralizar todas las operaciones en un único servidor dificultaba la capacidad de aislar problemas de seguridad, aumentando el impacto potencial de cualquier incidente.

Justificación de uso

La integración de **Auth0** en el proyecto tiene como objetivo principal la gestión de acceso seguro entre la aplicación Flutter y la API de Django. La elección de esta herramienta se basa en dos características fundamentales: la implementación simplificada de autenticación segura y la gestión de permisos y roles.

1. **Implementación simplificada de autenticación segura:** Auth0 ofrece flujos de autenticación listos para usar, basados en protocolos reconocidos como OAuth 2.0 y OpenID Connect. Estos protocolos permiten proteger la API mediante el uso de tokens de acceso, que garantizan que solo los usuarios autenticados puedan interactuar con los endpoints de la API. Esta implementación reduce la complejidad

de crear un sistema de autenticación desde cero, al mismo tiempo que asegura que las mejores prácticas de seguridad.

2. **Gestión de permisos y roles:** Una de las ventajas clave de Auth0 es su capacidad para implementar **control de acceso basado en roles (RBAC)**. Esta funcionalidad permite definir qué acciones o recursos dentro de la API pueden ser accedidos por cada usuario, según su rol específico. En el contexto de la aplicación Flutter, esta característica es particularmente útil, ya que se pueden establecer distintos niveles de permisos para diferentes tipos de usuarios, como administradores, moderadores o usuarios regulares

Flujo de comunicación

Con Auth0 el flujo de comunicación entre el cliente (app móvil) y el servidor (API Django) cambiaron, separando la lógica de negocio perteneciente a la API con la lógica de autenticación.

El flujo que ahora es el siguiente:

1. El cliente móvil autentica al usuario en Auth0.
2. Auth0 genera un JWT que es devuelto al cliente móvil.
3. El cliente móvil envía el JWT en el encabezado de cada solicitud a la API de Django.
4. Django valida el JWT utilizando las claves públicas de Auth0.
5. Django extrae los claims del JWT y utiliza los datos (como el identificador de usuario o el correo) para realizar acciones en la API.
6. La API responde al cliente móvil con los datos solicitados.

Este nuevo flujo se puede ver de manera simplificada en la siguiente figura:

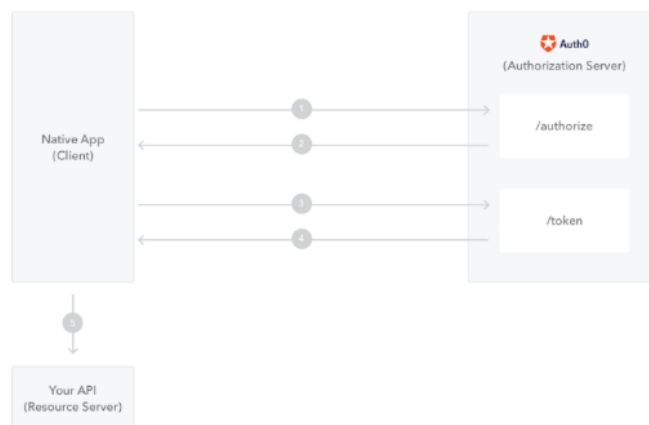


Figura 30: Comunicación entre cliente y API con seguridad aplicada

Fuente: *Solution Overview (Mobile Apps + API)*, Auth0 2024

Con la nueva implementación para que los usuarios, mediante el cliente móvil puedan acceder a su información a través la API, deben proporcionar el token de acceso que viene firmado con un algoritmo de firmado (HS256 o RS256), el cual es firmado por el servicio de Auth0. El firmado asegura que el token no fue modificado.

Luego de que el cliente obtiene el token de acceso firmado, puede enviar una petición segura a la API.

Cuando una petición llega a la API Django, el servidor decodifica el token, verifica la autenticidad del token (que la firma sea válida), y mediante el Payload (carga útil) del token, se identifica el usuario.

Sin embargo, si la API determina que la firma no es válida, o el tiempo de expiración paso la solicitud será denegada, devolviendo un código HTTP 401 (unauthorized)

Configuraciones necesarias

Configuraciones en API: Para que la API pueda manejar token, es decir, codificarlo y verificarlo, es necesario una librería de JWT para Django, como simple-jwt, que permite la manipulación simple de JSON Web Tokens, también es necesario para poder autenticar al usuario, usar jwt-drf, ya que se definen las clases de permiso, que permitirán autenticar a los usuarios mediante los tokens JWT, en la siguiente figura se muestra parte de las configuraciones importante en la API:

```
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'rest_framework_jwt.authentication.JSONWebTokenAuthentication',
        'rest_framework.authentication.SessionAuthentication',
        'rest_framework.authentication.BasicAuthentication',
    ),
    'DEFAULT_PERMISSION_CLASSES': (
        'rest_framework.permissions.IsAuthenticated', # hace que tenga que estar autenticado para poder hacer peticiones
    ),
}

AUTH0_DOMAIN = get_env_var('AUTH0_DOMAIN')
AUTH0_AUDIENCE = get_env_var('AUTH0_AUDIENCE')

JWT_AUTH = {
    'JWT_PAYLOAD_GET_USERNAME_HANDLER': 'api.utils.jwt_get_username_from_payload_handler',
    'JWT_DECODE_HANDLER': 'api.utils.jwt_decode_token',
    'JWT_ALGORITHM': 'RS256',
    'JWT_AUDIENCE': AUTH0_AUDIENCE,
    'JWT_ISSUER': f'https://{AUTH0_DOMAIN}/',
    'JWT_AUTH_HEADER_PREFIX': 'Bearer',
}
```

Figura 31: Configuraciones importantes de autenticación mediante JWT en Django

Fuente: Elaboración propia

Configuraciones en aplicación móvil (Flutter): En Flutter, también es necesario instalar dependencias para integrarse con Auth0. Además, se requiere configurar una clase de servicio que implemente los métodos `getAccessToken`, `login`, `signup` y `logout`. Parte de las configuraciones más relevantes se ilustran en la siguiente figura:

```
class AuthService {}
static final AuthService instance = AuthService._internal();
factory AuthService() => instance;

Auth0 auth0 =
  Auth0(dotenv.env['AUTH0_DOMAIN']!, dotenv.env['AUTH0_CLIENT_ID']!);
UserProfile? profile;
String? _accessToken;

AuthService._internal();

Future init() async {
  final isLoggedIn = await auth0.credentialsManager.isValidCredentials();
  if (isLoggedIn) {
    final credentials = await auth0.credentialsManager.credentials(
      parameters: {'audience': dotenv.env['AUTH0_AUDIENCE']!},
    );
    profile = credentials.user;
    _accessToken = credentials.accessToken;
  }
  return profile;
}

/// Método para obtener el accessToken
Future<String?> getAccessToken() async {
  if (_accessToken == null) {
    final credentials = await auth0.credentialsManager.credentials(
      parameters: {'audience': dotenv.env['AUTH0_AUDIENCE']!},
    );
    _accessToken =
      credentials.accessToken; // Actualiza el token si es necesario
  }
  return _accessToken;
}
```

Figura 32: Configuración de Flutter con Auth0.

Fuente: Elaboración propia.

Resumen autenticación y autorización

La solución propuesta, que utiliza el servicio Auth0, establece una comunicación segura entre la API y el cliente a través del envío de tokens de acceso. Estos tokens están codificados y firmados, lo que garantiza la integridad de los datos contenidos en ellos y permite verificar su autenticidad durante la transmisión.

Esta implementación del servicio Auth0 segmenta las responsabilidades y ofrece la ventaja de no gestionar directamente la seguridad del servidor de autenticación. Sin embargo, la configuración y las correctas implementaciones siguen siendo responsabilidad de los desarrolladores y del equipo de operaciones.

A pesar de ello, el uso de Auth0 ofrece un nivel de seguridad superior y reduce considerablemente el esfuerzo que tendría que invertir la organización para implementar una solución de autenticación propia.

4.2.5.2 Firewall de aplicaciones

Para la implementación de firewall de aplicaciones se utilizó el servicio de Cloudflare (WAF). Este capítulo abordará cómo se utilizaron las capacidades del WAF para filtrar, monitorear y bloquear el tráfico HTTP/HTTPS dirigido a la aplicación web, detallando el proceso de implementación y los beneficios que aportó a Not Waste.

Para explicar la implementación del firewall, primero se describirá el contexto previo a la integración del firewall de aplicaciones. Posteriormente, se detallará el proceso de integración del servicio, incluyendo el flujo de comunicación entre el servidor (API Django) y Cloudflare (WAF), así como la configuración de las reglas de seguridad aplicadas para filtrar las solicitudes de internet antes de que sean lleguen al servidor de Not Waste.

Para comprender mejor este capítulo, se recomienda revisar la sección sobre la identificación de servicios de seguridad, donde se dedica un apartado específico a la explicación del Firewall de Aplicaciones Web (WAF). Además, se sugiere consultar la sección de “detalle de servicios”, que ofrece una introducción más amplia sobre Cloudflare y su funcionamiento.

Contexto anterior a Cloudflare WAF

En el contexto anterior a la implementación de un firewall de aplicaciones (WAF), todas las solicitudes llegaban directamente al servidor, sin distinción entre peticiones legítimas de usuarios o solicitudes no autenticadas.

La aplicación solo contaba con usuarios de prueba, y aun así el servidor recibía muchas peticiones no identificadas, lo que significaba que muchas de las solicitudes no eran realizadas por usuarios reales, sino que algunas provenían de bots, los cuales fueron posteriormente identificados por Cloudflare. Esto resultaba en un tráfico no deseado hacia el servidor, lo que podría afectar negativamente su rendimiento, en la siguiente figura se mostrara el tráfico que ha pasado por el servidor y que fue mitigado, cabe destacar que un porcentaje de las peticiones venia de un servicio de monitoreo que se explicara más adelante, pero no todas las peticiones eran de este servicio:

Análisis de tráfico

Revise los picos en el tráfico HTTP y los atributos de solicitud superior.

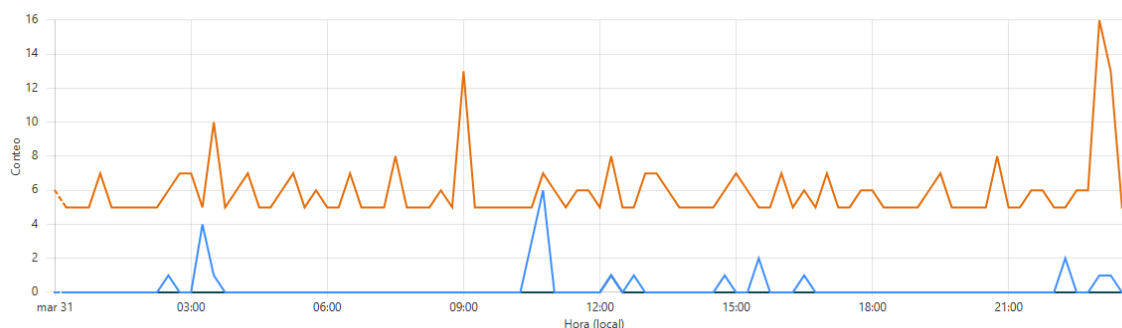
Todo País Navegador de origen Sistema operativo de origen Tipo de dispositivo de origen IP de origen ...

Total
592

Mitigado por WAF
566

Entregado por Cloudflare
1

Servidor por origen
25



La figura anterior representa un ejemplo de peticiones realizadas a la API (Django), donde se observa un total de 592, y 566 peticiones que no tienen los requisitos para llegar al servidor, es decir, no son autorizadas. Esta cantidad de peticiones pueden afectar los servicios de la organización, ya que, en un entorno de producción con usuarios reales, el tráfico innecesario podría poner en riesgo la infraestructura del servidor, puesto que consumiría recursos de manera ineficiente e incluso podría derivar en un ataque de denegación de servicio (DoS). El consumo excesivo e innecesario, puede afectar la disponibilidad del servidor al no tener suficientes recursos para responder a los usuarios que, si son legítimos, eso vulnera uno de los pilares fundamentales de la seguridad informática: la Disponibilidad (*Availability*), como se explicó en el Capítulo 2 sobre Introducción a la Ciberseguridad.

Justificación de uso

Anteriormente, todas las solicitudes llegaban al servidor (API) sin un criterio claro que diferenciara entre usuarios autorizados y no autorizados. Cloudflare presentó una solución efectiva al implementar un mecanismo para identificar y filtrar quién puede acceder al servidor.

Implementación de WAF

Para la implementación de Cloudflare como WAF, se debe seguir una serie de pasos de implementación, los cuales fueron llevados a cabo en Not Waste:

1. Cambiar los Nameservers.
2. Ingreso de Nombres de dominio (DNS).
3. Ajuste de Servidor web (Nginx) para recibir el tráfico desde el en nombre de dominio (DNS).
4. Configuración de certificados SSL en Nginx,

Para utilizar los servicios de Cloudflare, es necesario contar primero con un nombre de dominio, como por ejemplo "pepito.com". Existen diversas opciones para registrar un dominio, como Hostinger, GoDaddy o incluso Cloudflare. En el caso de estudio, se utiliza el dominio *notwaste.tech*.

El siguiente paso es registrar este dominio en Cloudflare y configurar sus Nameservers, para que el tráfico pase por servidores del servicio. De esta forma, cada vez que se realice una solicitud HTTP al dominio *notwaste.tech*, lo que significa que se quiere enviar una petición al servidor de API Django, el tráfico será dirigido a través de Cloudflare, que actuará como un proxy inverso.

La siguiente figura muestra cómo se establecieron los dominios y subdominios en los servidores de Cloudflare:

☐	Tipo ⓘ	Nombre ⓘ	Contenido ⓘ	Estado de proxy ⓘ	TTL ⓘ	Acciones
☐	A	api	IPv4	 Redirigido mediante proxy	Automático	Editar ▶
☐	A	notwaste.tech	IPv4	 Redirigido mediante proxy	Automático	Editar ▶

Figura 33: Registros DNS en Cloudflare.

Fuente: Elaboración propia.

Este servicio aplicará las reglas de seguridad definidas por el usuario y Cloudflare, permitiendo que solo las peticiones autorizadas (filtradas por las reglas de Cloudflare) lleguen al servidor web (NGINX).

Una de las ventajas de Cloudflare, es que el tráfico que pasa por sus servidores ya usa cifrado SSL/TLS, pero Cloudflare en un principio está configurado para ser flexible, lo que

significa que solo el tráfico se cifra entre Cloudflare y el Cliente que realiza la petición, pero no entre el servidor y Cloudflare, ya que esta opción debe configurarse en el servidor web.

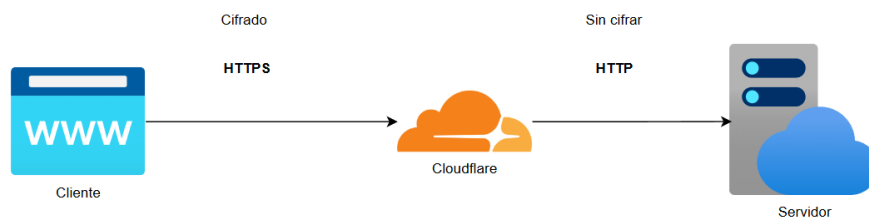


Figura 34: Comunicación cliente servidor predeterminado de Cloudflare
Fuente: Elaboración propia.

El Método predeterminado de Cloudflare no es recomendado, por ello se realizó una modificación para usar el método completo, que permite usar un cifrado también entre el servidor de origen y Cloudflare. Se implementó HTTPS en el servidor web (NGINX) con certificados auto firmados para habilitar HTTPS⁶. De esta manera todo el tráfico está cifrado. Tal como se muestra en la siguiente figura:

Cifrado SSL/TLS

Modo de cifrado actual: Completo

El modo de encriptación fue cambiado por última vez hace 7 días.

Modo automático deshabilitado hace 11 días.



Figura 35: Comunicación cliente servidor con Cloudflare, modo cifrado completo.
Fuente: Elaboración propia.

Flujo de comunicación

El flujo de comunicación descrito a continuación detalla cómo las solicitudes pasan a través de Cloudflare.

⁶ En el capítulo 4.5.3 Verificación, se verifica el funcionamiento del cifrado HTTPS

Cuando una solicitud atraviesa Cloudflare, esta es filtrada mediante un conjunto de reglas preconfiguradas y definidas por la plataforma (Cloudflare, 2024). Estas reglas incluyen protecciones contra ataques DDos y mecanismos de mitigación de bots, entre otras medidas de seguridad. Algunas de estas reglas son ajustables, lo que permite personalizarlas según necesidades específicas.

Definición de reglas del WAF

Las reglas de Cloudflare son configuraciones personalizables que permiten definir cómo se debe gestionar el tráfico entrante hacia un sitio web o aplicación. En el caso de Not Waste, la aplicación corresponde a la API de Django. Estas reglas permiten filtrar las peticiones entrantes, lo que posibilita bloquear aquellas que no cumplan con los criterios establecidos y permitir solo las solicitudes que coincidan con las configuraciones definidas.

La siguiente figura ilustra gráficamente cómo ciertas solicitudes pueden ser denegadas mediante reglas:

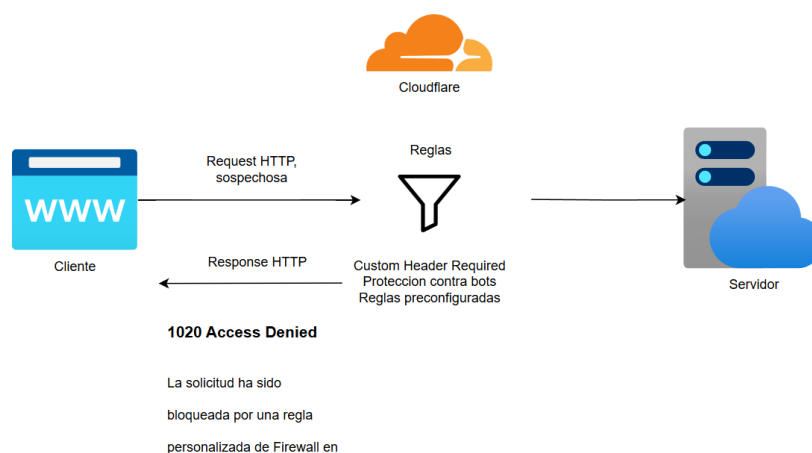


Figura 36: Reglas de acceso al servidor mediante Cloudflare.

Fuente: Elaboración propia.

En el WAF de Cloudflare en Not Waste, se definieron dos reglas personalizadas: “*custom header required*” y “*protección de bots*”.

+ Crear regla		Q Buscar...		Buscar	Mostrar filtros
Orden	Acción	Nombre	CSR ⓘ	Actividad en las últimas 24 horas	Habilitado
1	Bloquear	custom header required Encabezado	-	 623	☒
2	Bloquear	Protección Bots Bots conocidos	-	 0	☒

Figura 37: Reglas personalizadas, WAF Cloudflare
Fuente: Elaboración propia

La regla número uno “*custom header required*” verifica que las solicitudes incluyan un encabezado personalizado con un valor específico, como MiRhtJ4dDEW5cd. Si la cabecera de la petición no contiene este valor, la solicitud es rechazada. Esto ayuda a filtrar y permitir únicamente las solicitudes que cumplan con este requisito, evitando que cualquier petición no autorizada alcance el servidor web. De esta manera el servidor web, no se saturará por peticiones indeseadas. Sin embargo, esto no autentica ni autoriza al usuario, la autenticación del usuario se realiza en la API Django utilizando un token de acceso (JWT).

La Figura X.1 y X.2 muestran mitigaciones realizadas tanto por las reglas personalizadas “*Custom Header required*” y por una regla predeterminada por Cloudflare “modo bot fight”.

Registros muestreados

Exportar

Editar columnas

Fecha	Acción realizada	País	Dirección IP	Servicio
1 de ene. de 2025 18:41:58	Bloquear		IP	Reglas personalizadas
1 de ene. de 2025 18:41:26	Bloquear		IP	Reglas personalizadas

Servicio coincidente

Exportar evento JSON

ServicioReglas personalizadas

Acción realizadaBloquear

Conjunto de reglasdefault
...b18eda73

Regla
custom header required
...e1c5143a

Detalles de la solicitud

Id. de rayo8fb5ac6a3c7aa777

Dirección IP

ASN

País

Agente de usuarioPostmanRuntime/7.43.0

Versión HTTPHTTP/1.1

ReferenteNinguno (directo)

MétodoGET

Servidorapi.notwaste.tech

Ruta

Cadena de consultaCadena de consulta vacía

Figura 38: Mitigación regla personalizada Header required, Cloudflare WAF
Fuente: Elaboración propia

1 de ene. de 2025 18:23:06
Desafío manejado

Modo bot fight

Servicio coincidente

Servicio

Modo bot fight

Acción realizada

Desafío manejado

[Exportar evento JSON](#)

Detalles de la solicitud

Id. de rayo

8fb591900b23971e

Referente

Ninguno (directo)

Dirección IP

Método

GET

ASN

Servidor

notwaste.tech

País

Ruta

/

Agente de usuario

Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/117.0

Cadena de consulta

Cadena de consulta vacía

Versión HTTP

HTTP/1.1

Figura 39: Mitigación regla predeterminada Modo bot figth, Cloudflare WAF

Fuente: Elaboración propia

Resumen de Firewall

Se logró mitigar solicitudes no autorizadas al sistema provenientes de bots y actores no identificados. Un ejemplo claro de esto ocurrió cuando se mostraron las definiciones de reglas del firewall que permitieron bloquear peticiones que no contenían una custom key en el Header, lo que significaba que las peticiones no eran de los clientes, otro ejemplo es el bloqueo de bots conocidos por Cloudflare.

Estas medidas redujeron significativamente el riesgo de que solicitudes no autorizadas inundaran la red o intentaran acceder a recursos restringidos.

Además de ofrecer un servicio de firewall de aplicaciones, Cloudflare proporciona herramientas adicionales que facilitan la revisión de registros y métricas del tráfico web. Estas herramientas permitieron obtener un panorama detallado sobre el uso que está recibiendo la aplicación, como la cantidad de solicitudes, la distribución geográfica de los usuarios, las peticiones bloqueadas y las amenazas detectadas.

Adicionalmente, en las etapas iniciales, cuando aún no se había implementado una comunicación segura entre Internet y el servidor de la API, Cloudflare proporcionó una capa adicional de seguridad al agregar automáticamente a la comunicación del servidor con los clientes mediante TLS (*Transport Layer Security*) que cifra los datos que se transmiten a la comunicación, garantizando la protección de los datos en tránsito.

4.2.5.3 Monitoreo y Registro

Al igual que en la implementación de otros servicios, para comprender mejor la implementación de "Monitoreo y Registro", se recomienda explorar el proceso de identificación de servicios de seguridad, que proporciona una introducción a los conceptos clave de monitoreo y registro.

El registro y monitoreo se llevaron a cabo utilizando el servicio **Better Stack**, que centraliza los logs provenientes de las fuentes establecidas, como la API de Django y el servidor Linux. Para tener más información del servicio se recomienda revisar detalle de servicios.

El servicio fue seleccionado por su facilidad de integración con las tecnologías empleadas en Not Waste, como Django, Linux y Nginx. Además, su extensa documentación, que incluye ejemplos claros y específicos para estas tecnologías, representó una ventaja significativa. Este aspecto, junto con su simplicidad de uso, fue un factor clave para preferirlo sobre otras opciones de monitoreo y registro, como CloudWatch de AWS.

Contexto anterior a los registros

Antes de implementar el uso de registros, se desconocía el estado del servidor, se desconocía si el servidor estaba recibiendo peticiones. Dado que la aplicación móvil aún no estaba disponible para su uso público, se asumía que no había solicitudes. Como resultado, no se tenía conocimiento sobre el tipo de peticiones que se realizaban, lo que dificultaba la detección de posibles vulnerabilidades debido a configuraciones erróneas. Además, no se podía identificar el tipo de peticiones sospechosas que podrían haberse realizado.

Identificación de sistemas para obtener registros (logs)

Las actividades que se realizaron para comenzar a monitorear los sistemas, es la obtención de registros de los activos tecnológicos de Not Waste, como registros del servidor Linux con RSlog y Registros provenientes de Django

A continuación, se detallan las actividades usadas para monitorear la infraestructura de Not Waste:

- Identificar los sistemas/tecnologías de los cuales desea tener un registro de acceso
- Definir los eventos que va a eventos registrar
- Establece un directorio o ubicación centralizada donde se almacenarán los registros
- Asegúrate de que la ubicación está protegida contra accesos no autorizados

En la arquitectura de Not Waste, se identificaron varios sistemas posibles para ser monitoreados. De estos, se seleccionaron para su monitoreo el firewall de aplicaciones (WAF), el servidor Linux y la API (Django). La siguiente tabla muestra el alcance de los sistemas que serán monitoreados:

Tabla 10: Alcance de monitoreo.

Fuente: Elaboración propia.

Sistemas	Monitoreo
Base de datos	No
WAF	Si
Servidor Linux	Si
Servicio de nube	No
API (Django)	Si
IDS	No
IPS	No
Servidor Web (Nginx)	No

Es importante señalar que el monitoreo del WAF no se presenta en esta sección, ya que su implementación fue abordada previamente en la sección correspondiente al firewall de aplicaciones.

Introducción a los registros

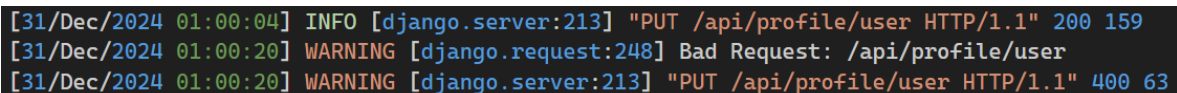
El manejo de registros es un ámbito amplio y fundamental en el desarrollo de software. Va más allá de simplemente almacenar mensajes en la consola o en un archivo; permite construir sistemas avanzados y robustos que integran y gestionan registros provenientes de múltiples fuentes. Estos sistemas pueden recopilar datos desde aplicaciones, servidores, bases de datos, etc. Consolidándolos en un único lugar para facilitar su análisis y monitoreo.

Además, los registros pueden ser procesados en tiempo real para detectar anomalías, rastrear eventos críticos, identificar patrones de uso o incluso responder automáticamente ante ciertas condiciones, como alertar a los administradores en caso de errores graves o intentos de acceso no autorizado.

Contexto de registros en Python

Los registros (log) en Python, son similares a las funciones “print”, donde al realizar una acción que desencadena la función “print”, se mostrara por la terminal (cmd, bash o variantes) la información contenida en el “print”.

Los registros son mucho más potentes y flexibles, ya que permiten categorizar, formatear y dirigir los mensajes a diferentes destinos (consola, archivos, servicios externos, etc.), mientras que los print solo muestran texto en la consola. Un ejemplo de registro serían los que se muestran en la siguiente figura:



```
[31/Dec/2024 01:00:04] INFO [django.server:213] "PUT /api/profile/user HTTP/1.1" 200 159
[31/Dec/2024 01:00:20] WARNING [django.request:248] Bad Request: /api/profile/user
[31/Dec/2024 01:00:20] WARNING [django.server:213] "PUT /api/profile/user HTTP/1.1" 400 63
```

Figura 40: Ejemplo registros con Django.

Fuente: Elaboración propia.

En Not Waste la API está construida en el framework de Django. Para realizar el manejo de los registros, Django utiliza y amplía el módulo incorporado de Python logging para realizar el registro del sistema. Los siguientes conceptos son componentes clave en el sistema de logging de Python:

- *Logger*: Es el objeto que crea los mensajes.
- *Handler*: Decide dónde enviar los mensajes (archivo, consola, etc.).
- *Filter*: Permite incluir o excluir ciertos mensajes.

- *Formatter*: Define el formato de los mensajes.

El sistema de logging clasifica los mensajes en diferentes niveles según su importancia o gravedad. Estos niveles permiten controlar qué mensajes deben ser capturados y procesados por los *loggers* y *handlers*:

- **DEBUG**: Mensajes detallados utilizados para diagnosticar problemas durante el desarrollo.
- **INFO**: Mensajes que confirman el funcionamiento esperado del sistema.
- **WARNING**: Avisos sobre situaciones inesperadas o potencialmente problemáticas que no impiden el funcionamiento del sistema.
- **ERROR**: Mensajes que indican errores que afectan la funcionalidad de la aplicación.
- **CRITICAL**: Mensajes que reflejan errores graves que podrían interrumpir completamente el funcionamiento del sistema.

Implementación registros con Django y Better Stack

Para comenzar a utilizar los registros para monitorear los sistemas en la API de Not Waste, se siguen los siguientes pasos: elegir qué datos generar y qué parte del código hacerlo, cómo formatear los registros, dónde transmitir los registros.

En Django se puede configurar los registros (log) en el archivo de configuración del proyecto "settings.py", en este archivo se puede definir un diccionario que contenga la se puede definir diferentes "handlers" para enviar los registros a distintos destinos (archivo, consola, etc.), y "loggers" para decidir qué tipos de registros capturar.

Elegir que datos generar: En Not Waste, los datos que se desean capturar son los mensajes que tiene un nivel superior a WARNING, es decir, los de INFO y DEBUG, que tienen niveles más bajos de WARNING no se desean capturar.

Como formatear los registros: los *formatters* sirven para definir la estructura en la que se quiere que se guarden los registros. Para darle un formato a los registros (logs) se usan distintos marcadores o *placeholders* que definen una parte del registro, en la siguiente tabla se muestran los marcadores que se usaron para generar el formato de los registros en la API de Not Waste:

Tabla 11: Formato de logs para API Django.

Fuente: Elaboración propia.

Formateo usado para logs de API Django		
Marcador	Descripción	Ejemplo
%(asctime)s	Fecha y hora en que se generó el mensaje de log.	2025-01-02 15:30:45,123
%(levelname)s	Nivel de severidad del mensaje de log.	WARNING, ERROR, CRITICAL
%(name)s	Nombre del logger que generó el mensaje.	django.request
%(lineno)s	Número de línea en el archivo de código donde se generó el mensaje.	213
%(message)s	Mensaje principal del log, proporcionado al llamar al logger.	Este es un mensaje de prueba

El siguiente paso es definir donde enviar los registros, para ello en Django se usan los Handler, en el proyecto de Not Waste hay dos definidos.

- Handler (file_django): envía datos de registros a un archivo “django.log” el cual guarda datos de nivel INFO, los registros del archivo “django.log” se almacenan así:

```
[31/Dec/2024 10:47:57] INFO [django.server:213] "GET /api/discarded_money_food/ HTTP/1.1" 200 28
[31/Dec/2024 10:47:57] INFO [django.server:213] "GET /api/near_expiration/ HTTP/1.1" 200 2
[31/Dec/2024 10:54:09] WARNING [django.request:248] Not Found: /api/food/user/
[31/Dec/2024 10:54:09] WARNING [django.server:213] "GET /api/food/user/ HTTP/1.1" 404 23
[31/Dec/2024 10:54:19] INFO [django.server:213] "GET /api/food/user HTTP/1.1" 200 1824
```

Figura 41: Registros de API Django.

Fuente: Elaboración propia.

- Handler (logtail): envía datos de registros a el servicio centralizado “logtail”, en logtail los logs se formatean de distinta manera, ya que el este *Handler* viene preconfigurado cuando se instala como nueva dependencia en Django y se enviar al servidor centralizado de BetterStack, los logs enviados finalmente se ven de la siguiente manera en el servicio:

```
> 2025-01-08 20:31:13.825 [django_test] [WARNING] 1074 139077742731392 ../lib/python3.12/site-packages/django/utils/log.py 248 Ba
v 2025-01-08 20:31:17.486 [django_test] [ERROR] 1074 139077742731392 ../lib/python3.12/site-packages/django/core/handlers/exception.py 124 :
  context
    runtime
      "file": "../lib/python3.12/site-packages/django/core/handlers/exception.py"
      "function": "response_for_exception"
      "line": 124
      "logger_name": "django.security.DisallowedHost"
      "thread_id": 139077742731392
      "thread_name": "MainThread"
    system
      "pid": 1074
      "process_name": "MainProcess"
  "dt": "2025-01-08T20:31:17.486517+00:00"
  "filename": "exception.py"
  "level": "error"
  "message": "Invalid HTTP_HOST header: '52.70.179.199'. You may need to add '52.70.179.199' to ALLOWED_HOSTS. Traceback (most recent call last): Fil
  "severity": 4
  "status_code": 400
  Collapse all Copy JSON
2025-01-08 20:31:17.488 [django_test] [WARNING] 1074 139077742731392 ../lib/python3.12/site-packages/django/utils/log.py 248 Bad Request:
```

Figura 42: Registros API Django en Better Stack.

Fuente: Elaboración propia.

Resumen de utilización de registros

Se facilitó la revisión de los logs, evitando la necesidad de conectarse directamente al servidor para acceder a ellos. Esto hizo que la revisión de logs fuera más común y estructurada, proporcionando una estructura más útil y accesible para su análisis.

Además, permitió consultar registros provenientes de diversas fuentes de manera más sencilla. Gracias a la identificación de posibles vulnerabilidades, se implementaron medidas adicionales, como la revisión de configuraciones para garantizar que solo se aceptaran solicitudes que pasaran a través de Cloudflare.

4.3 Práctica RT-CV-1

En la práctica RT-CV-1 se llevará a cabo la explicación e implementación del objetivo definido en los pasos previos, específicamente "Prueba de controles de seguridad del software". La práctica establecida para el cumplimiento del objetivo establecido es RT-CV-1, de acuerdo con el modelo de madurez SAMM, la práctica *Requirements-driven Testing* (RT) tiene como propósito asegurar que los controles de seguridad implementados funcionen de manera adecuada y cumplan con los requisitos de seguridad establecidos en el proyecto. En particular, dentro del flujo *Control Verification* (CV), correspondiente al nivel de madurez 1 (CV-1), se busca realizar pruebas de seguridad para confirmar que los controles de seguridad estándar del software operen conforme a lo esperado.

Explicación de la actividad: Como lo menciona el marco de seguridad, lo que debe realizarse es probar el correcto funcionamiento de los controles de confidencialidad, integridad y disponibilidad de los datos, así como del servicio. Mediante pruebas de seguridad que incluyen, al menos, pruebas de autenticación, control de acceso, validación de entrada, codificación y escape de datos y controles de cifrado.

Criterios de calidad: Los criterios de calidad que debe considerarse según SAMM, para poder cumplir de manera efectiva los requisitos de seguridad que se establece para esta práctica son:

- Las pruebas de seguridad al menos verifican la implementación de autenticación, control de acceso, validación de entrada, codificación y escape de datos y controles de cifrado.
- Las pruebas de seguridad se ejecutan siempre que la aplicación cambia su uso de los controles.

Al finalizar esta práctica se evaluará el cumplimiento de los criterios de calidad establecido anteriormente. Es necesario definir que conforme a lo señalado en los proceso de planificación de SAMM, se determinó que el alcance de la realización de las prácticas comprende a la API Django.

4.3.1 Proceso

El proceso para llevar a cabo esta actividad comienza con la identificación de una guía de referencia que proporcione información detallada sobre los controles específicos relacionados con las pruebas de seguridad definidas en la práctica RT-CV-1. Estos controles incluyen autenticación, control de acceso, validación de entradas, escape de datos y cifrado. Posteriormente, se identificaron los controles de seguridad derivados de la guía de referencia seleccionada y se concretaron en casos de prueba, los cuales se utilizaron para realizar las pruebas establecidas. La siguiente figura muestra un diagrama del proceso explicado.

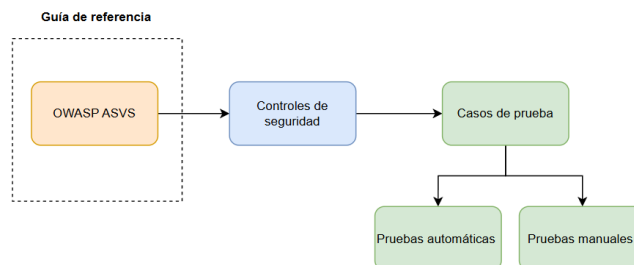


Figura 43: Proceso Creación de pruebas en API

Fuente: Elaboración propia.

Para diseñar las pruebas de seguridad en la API se utilizó como referencia OWASP ASVS versión 4.0.3, que es un estándar que define una serie de controles de seguridad de aplicaciones web organizados en 14 categorías, tales como autenticación, control de acceso, validación de entradas, cifrado, configuración, entre otros. Los controles dentro de cada categoría están estructurados en tres niveles:

- Nivel 1: Controles básicos, enfocados en mitigar las vulnerabilidades más comunes y garantizar una seguridad mínima adecuada para todas las aplicaciones.
- Nivel 2: Controles avanzados, que abordan riesgos más complejos y son esenciales para aplicaciones que manejan datos sensibles o tienen un mayor riesgo de ser atacadas.
- Nivel 3: Controles exhaustivos, diseñados para aplicaciones críticas donde se requiere la máxima seguridad, como en sectores financieros, gubernamentales o de salud.

Una vez seleccionado el estándar OWASP ASVS como guía para el desarrollo de casos de prueba, es necesario llevar a cabo una evaluación detallada de los controles que ofrece. Dado que el estándar presenta una variedad de controles organizados en niveles, es

importante priorizar aquellos que deben evaluarse primero, especialmente en organizaciones pequeñas como Not Waste. En este caso, se optó por analizar los controles del nivel 1, que son los más básicos y fundamentales.

Además, al realizar la evaluación de software, es crucial identificar qué controles son aplicables según las características de la solución informática en cuestión. Dependiendo del diseño del software, algunos controles pueden no ser relevantes. Por ejemplo, en el caso de la API de Not Waste, que no tiene un Frontend web, los controles relacionados con la protección contra ataques de tipo CSRF (Cross-Site Request Forgery) no son necesarios, ya que este tipo de vulnerabilidad es más común en aplicaciones web tradicionales que manejan formularios y sesiones autenticadas a través de navegadores. En el caso de Not Waste, este control no es relevante porque el flujo de interacción no utiliza un entorno web que dependa de estas medidas.

A partir de los controles de seguridad aplicables al diseño de la solución de Not Waste (específicamente la comunicación entre la API y Not Waste), se crearon los casos de prueba para verificar el cumplimiento de los controles de seguridad del OWASP ASVS, y posteriormente se realizaron los test utilizando dos herramientas específicas, Postman para pruebas manuales y Pytest para pruebas automáticas.

4.3.2 Conocimiento previo

Para comprender y aplicar las actividades requeridas en la práctica RT-CV-1, es necesario contar con ciertos conocimientos previos. Entre estos, se incluyen una comprensión básica sobre qué son las pruebas (tests), los tipos más comunes de pruebas, qué son los controles y los estándares de verificación.

Sin embargo, dado que los estándares de verificación ya fueron introducidos en el Capítulo 2: Marco Metodológico, no se abordarán nuevamente en la explicación de los contextos necesarios para esta práctica.

4.3.3 Pruebas de software

Las pruebas de software o *testing (IBM)* son el proceso de evaluar y verificar que un producto o aplicación de software hace lo que se supone que debe hacer. Entre los beneficios de unas buenas pruebas se incluyen la prevención de errores, la mejora del rendimiento y la verificación de controles de seguridad.

Diferentes criterios permiten clasificar las pruebas de software en función de su enfoque, alcance y objetivo. A continuación, se presenta una breve explicación de los tipos de prueba, que brindan una visión general sobre cómo se pueden abordar las pruebas técnicas.

Tabla 12: Tipos de prueba.
Fuente: Elaboración propia.

CRITERIO	TIPOS DE PRUEBA EXISTENTES		
FACTOR DE VALIDACIÓN (OBJETIVO)	Funcional	No funcional	
AUTOMATIZACIÓN	Manual	Automática	
GRANULARIDAD	Unitaria	Integración	Sistemas

En base a la tabla anterior se ofrece un resumen de los tipos de pruebas basado en la información proporcionada por Atlassian en su comparación de pruebas de software “Los distintos tipos de pruebas de software” (Pittet, n.d.), que abarca la explicación y comparación de las pruebas automáticas, manuales, unitaria y de integración:

Prueba por Automatización

- Pruebas automatizadas: utilizan scripts predefinidos que se ejecutan de manera autónoma por una máquina. Estas pruebas son más rápidas, consistentes y menos

propensas a errores, lo que las hace ideales para escenarios repetitivos o de gran escala.

- Pruebas manuales: implican la interacción directa de un tester con el software, navegando por la interfaz o utilizando herramientas específicas para probar APIs. Aunque son útiles para pruebas exploratorias y evaluaciones de usabilidad, son costosas, lentas y propensas a errores humanos, ya que dependen de la ejecución manual de cada paso.

Pruebas por Granularidad

La clasificación de las pruebas por granularidad permite abordar distintos niveles de evaluación del software, desde componentes individuales hasta la interacción entre múltiples módulos. A continuación, se describen los principales tipos de pruebas según este criterio:

- Pruebas Unitarias: Las pruebas unitarias se caracterizan por su bajo nivel de abstracción, ya que se realizan directamente sobre la fuente de la aplicación. Estas pruebas tienen como objetivo verificar el correcto funcionamiento de métodos, funciones, clases o módulos individuales que componen el software. Su enfoque es validar de manera aislada la lógica implementada en cada unidad del sistema.
- Pruebas de Integración: Las pruebas de integración tienen como propósito asegurar que los distintos módulos o servicios que conforman la aplicación interactúan correctamente entre sí. Este tipo de pruebas abarca casos como la interacción con bases de datos, la comunicación entre microservicios o la integración de API externas. Dado que requieren que varias partes de la aplicación estén operativas, su ejecución suele ser más compleja y costosa en comparación con las pruebas unitarias.

Casos de prueba y pruebas de software

Una distinción importante en la práctica de seguridad RT-CV-1 es la diferencia entre los casos de prueba y los controles de seguridad. Mientras que los casos de prueba definen cómo deben verificarse los aspectos establecidos por los controles, los controles de seguridad especifican qué aspectos deben ser evaluados. Esta diferencia queda más claramente ilustrada en la Tabla 8.

Tabla 13: Diferencia entre control de seguridad y caso de prueba

Fuente: Elaboración propia.

	Control de seguridad	Caso de prueba
Definición	Son criterios o requisitos de seguridad que deben cumplirse.	Descripción de pruebas específicas diseñadas para validar la implementación de los controles.
Ejemplo	Verificar que los encabezados HTTP o cualquier parte de la respuesta HTTP no expongan información detallada sobre las versiones de los componentes del sistema.	Realizar solicitud GET en un endpoint y validar que la cabecera X-Powered-By no se encuentra en la respuesta, y el Header Server no incluye la versión del sistema.
Resumen	Define qué verificar	Define cómo verificar

4.3.4 Herramientas

Para crear y ejecutar pruebas de software de manera efectiva, es necesario contar con una serie de herramientas especializadas que permitan verificar la funcionalidad y seguridad del código. Las herramientas seleccionadas para llevar a cabo estas pruebas dependen de varios factores, como los objetivos específicos de la prueba, los tipos de pruebas que se desean realizar (unitarias, de integración, manuales, entre otras) y las tecnologías utilizadas en el desarrollo del proyecto. A continuación, se presenta una breve tabla con algunas de las opciones disponibles para realizar pruebas automáticas dependiendo del lenguaje de programación empleado en el software que se desea testear.

Tabla 14: Framework de pruebas por Lenguaje de programación.

Fuente: Elaboración propia.

Lenguaje	Framework de pruebas
Python	Pytest
Java	JUnit
JavaScript	Jest
Ruby	RSpec
PHP	PHPUnit
C#	NUnit

En el proyecto Not Waste, la organización ha desarrollado dos componentes principales: una aplicación móvil creada con Flutter y una API implementada con Django. Dentro del alcance definido para la práctica RT-CV-1, que se centra específicamente en las pruebas de la API de Django, se han utilizado dos herramientas principales: Pytest para pruebas automatizadas y Postman para pruebas manuales. A continuación, se detalla más información sobre ambas herramientas y su uso en este contexto.

Pytest: Es un framework de pruebas para Python que simplifica la escritura y ejecución de diversos tipos de pruebas de software, tales como pruebas unitarias e integración. Facilita la automatización del proceso de prueba mediante la escritura de código, lo que optimiza la verificación de la funcionalidad del software y garantiza su correcto funcionamiento de manera eficiente y repetible.

Pytest (Pytest, n.d.) describe la división de las pruebas en cuatro pasos fundamentales: Organizar, Actuar, Afirmar y Limpiar.

1. Organizar: Se enfoca en preparar el entorno de prueba. Incluye tareas como la creación de objetos, la inicialización de servicios, o la inserción de datos necesarios. Su objetivo es asegurar que todos los componentes estén listos para ejecutar la acción que se va a probar.
2. Actuar: Se realiza la acción que modificará el estado del sistema bajo prueba (SUT). Esta acción desencadena el comportamiento que se desea verificar, generalmente a través de una llamada a una función o método.
3. Afirmar: En este paso, se observa el estado resultante después de ejecutar la acción y se verifica que coincida con lo esperado. Las aserciones se utilizan para validar si el resultado obtenido es el correcto.

4. Limpiar: Este paso garantiza que el entorno de prueba se restaure a su estado original, evitando que las pruebas posteriores se vean influenciadas por cambios realizados durante la prueba actual.

Un ejemplo de una prueba desarrollada con Pytest se muestra en la siguiente figura. En esta prueba, se utiliza la instrucción `assert` para verificar que la respuesta obtenida coincide con la respuesta esperada, asegurando así que el comportamiento del código es el correcto:

```
# funcion.py

def suma(a, b):
    return a + b

# test.py

import pytest

def test_suma():
    # Test de suma positiva
    assert suma(2, 3) == 5

    # Test de suma negativa
    assert suma(-1, -1) == -2

    # Test con cero
    assert suma(0, 5) == 5

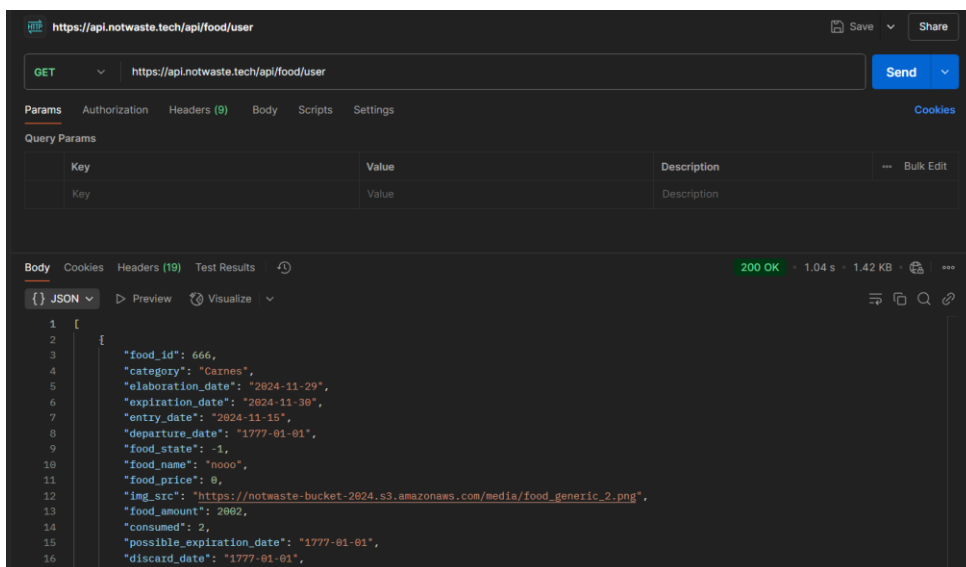
    # Test mixto
    assert suma(-3, 3) == 0
```

Figura 44: Ejemplo código Pytest.
Fuente: Elaboración propia.

Postman: Como se describe en la documentación de Postman (Postman, n.d.) es un sistema de software que reúne herramientas y procesos necesarios para trabajar con APIs a lo largo de su ciclo de vida completo, desde su diseño inicial hasta su implementación en producción y su posterior monitoreo.

En el caso de una API, Postman actúa como una herramienta para probar y consumir sus servicios de manera eficiente. Permite enviar solicitudes HTTP personalizadas, analizar las respuestas obtenidas y verificar que los datos devueltos cumplan con los requisitos establecidos. Esto es especialmente útil, ya que ayuda a identificar errores, validar la estructura y el contenido de las respuestas.

Para comprender mejor el uso de esta herramienta, se presenta la siguiente figura, que ilustra cómo, al realizar una petición a un endpoint, se recibe una respuesta que contiene información relevante para verificar los controles de seguridad. Esto incluye tanto las cabeceras como el contenido de las respuestas asociadas a la solicitud.



4.3.5 Implementación

Se presentarán los controles de verificación implementados para garantizar el cumplimiento de los requisitos de seguridad en la API Django de Not Waste.

Los controles seleccionados del estándar de verificación de aplicaciones ASVS, para los cuales se desarrolló un caso de prueba, se detallan en la tabla X. Como se puede observar, los casos de prueba corresponden a las siguientes categorías: control de acceso, validación de entradas, manejo de errores, comunicación, configuración, y recursos y archivos.

ID	Categoría	REF	ID REF	Realizado	Éxito	Tipo test
AC-1	Control de acceso	ASVS 4.0.3	V4.1.1	Si	Si	unitario
AC-7	Control de acceso	ASVS 4.0.3	V4.1.8	Si	Si	manual
AC-8	Control de acceso	ASVS 4.0.3	V4.1.3	No	-	-
AC-9	Control de acceso	ASVS 4.0.3	V4.2.1	No	-	-
IV-1	Validación de entradas	ASVS 4.0.3	V5.1.2	Si	Si	unitario
IV-5	Validación de entradas	ASVS 4.0.3	V5.1.4	Si	Si	manual
EH-1	Manejo de errores	ASVS 4.0.3	V7.4.1	Si	No	manual
CF-1	Configuración	ASVS 4.0.3	V14.4.1	Si	Si	unitario
CF-2	Configuración	ASVS 4.0.3	V14.3.2	Si	Si	manual
CF-3	Configuración	ASVS 4.0.3	V14.3.3	Si	Si	manual
CF-4	Configuración	ASVS 4.0.3	V14.4.4	Si	Si	manual
CF-5	Configuración	ASVS 4.0.3	V14.4.5	Si	Si	manual
FR-1	Archivos y recursos	ASVS 4.0.3	V12.1.1	No	-	-

La tabla anterior resume la información de las pruebas relacionadas con los controles de seguridad. A continuación, se presenta una breve explicación de los elementos que componen cada registro:

- **ID:** Identificador único de cada prueba. En este caso, cada código representa un caso de prueba, cuyo identificador se ha creado en función de su categoría. Por ejemplo, "AC-1" corresponde a "Access Control".
- **Categoría:** La categoría o área a la que pertenece la prueba. Por ejemplo, "Control de acceso" y "Validación de entradas" son categorías que agrupan diferentes controles de seguridad.
- **REF:** Referencia de los controles que se requieren verificar, en este caso en base a los controles del estándar ASVS 4.0.3.
- **ID REF:** Es el identificador de referencia dentro del estándar seleccionado. Por ejemplo, "V4.1.1", "V5.1.2" son referencias a secciones del estándar ASVS que describen controles específicos.
- **Realizado:** Indica si la prueba del caso de prueba fue realizada ("Si" o "No").
- **Éxito:** Indica si el caso de prueba se ejecutó con los resultados esperados ("Sí") o no tuvo los resultados previstos ("No"). Si no se ha realizado el caso de prueba, pero está definido, se indicará con un guion ("-").
- **Tipo test:** Especifica el tipo de prueba que se ha realizado. En la tabla, todas las pruebas realizadas son de tipo "unitario", lo que significa que son pruebas específicas de una unidad de código o función del sistema.

4.3.5.1 Entorno y herramientas de prueba

Para la ejecución de pruebas, es esencial contar con información detallada sobre el entorno que será evaluado, incluyendo las tecnologías y versiones presentes, así como las librerías más relevantes, especialmente en el caso de la API. La siguiente tabla presenta la información correspondiente a las herramientas utilizadas para realizar las pruebas, junto con las versiones de del sistema evaluado.

Tabla 15: Detalle de entorno de prueba.

Fuente: Elaboración propia.

Categoría	Detalles
Información del Servidor	
API	Python 3.12.3, Django 5.1.4, Gunicorn 23.0.0
Base de Datos	PostgreSQL 16.4
Sistema Operativo	Linux Ubuntu 24.0
Servidor Web	Nginx 1.24.0 (Ubuntu)
Herramientas de Prueba	
Framework de Pruebas	Pytest-django 4.9.0
Cliente de API	Postman (Windows) 11.23.3

4.3.5.2 Casos de prueba

A continuación, se describen los casos de prueba realizados junto con los tests asociados. Para cada caso de prueba, se detalla la forma en que debe ejecutarse la prueba y, posteriormente, se presentan los resultados y detalles correspondientes.

Los casos de prueba se organizan en una tabla que incluye el **ID del caso de prueba** y el **ID de referencia (ID REF)**, asociado al control específico según la guía utilizada. En este contexto, todas las referencias se basan en el estándar **OWASP ASVS 4.0.3**, seleccionado para evaluar los controles de seguridad de la API.

Tras cada caso de prueba, se incluye una explicación detallada de la prueba, acompañada de una imagen que ilustra su ejecución, en caso de ser necesario. Algunos casos de prueba no han sido probados, por ello en el resumen de los casos de prueba o en la tabla del caso de prueba aparecerá como pendiente.

Es importante destacar que estos casos de prueba no garantizan una validación completa de cada control específico. En algunos escenarios, es necesario realizar pruebas adicionales o definir nuevos casos de prueba para verificar plenamente ciertos controles.

Control de acceso

Tabla 16: Caso de prueba AC-1.

Fuente: Elaboración propia.

Caso de prueba de API			
ID	AC-1	Categoría	Control de acceso
REF	OWASP ASVS	ID REF	V4.1.1
Prerrequisitos	...		
Configuración requerida	Header sin token de acceso		
Método HTTP	GET	Endpoint	api/protected-message
Headers			
Descripción			
Realiza solicitudes con un usuario no autenticado (sin token) y asegurarse de que la API no permita acceso			
Resultados esperados	401 sin autorización		
Resultado actual	401 sin autorización		
Éxito/Fallo	Éxito	Realizado/Pendiente	Realizado
Notas	Para más seguridad se debe replicar en todos los Endpoint		

Como se detalla en la tabla, este caso de prueba se desarrolló mediante una prueba unitaria en Pytest. En ella, se realiza una solicitud sin el token de acceso, lo que resulta en la recepción de un código de respuesta 401 (No autorizado). Esto se debe a que la API requiere tokens de acceso para autenticar y procesar solicitudes entrantes. Este comportamiento asegura el cumplimiento del control V4.1.1, que establece que "la aplicación debe aplicar reglas de control de acceso en una capa de servicio confiable, especialmente cuando exista control de acceso del lado del cliente que podría ser eludido".

```

class TestProtectedMessageAPIView:
    """Tests para el endpoint de mensajes protegidos."""
    @pytest.mark.django_db
    def test_protected_api_view_without_token_returns_unauthorized(self, client):
        response = client.get(reverse('protected-message'))
        assert response.status_code == 401
        assert response.json() == {"detail": "Authentication credentials were not provided."}

    @pytest.mark.django_db
    def test_protected_api_view_with_invalid_token_returns_unauthorized(self, client):
        response = client.get(
            reverse('protected-message'), HTTP_AUTHORIZATION="Bearer invalid-token"
        )
        assert response.status_code == 401
        assert response.json() == {"detail": "Error decoding token."}

```

Figura 45: Prueba AC-1.

Fuente: Elaboración propia.

Tabla 17: Caso de prueba AC-7.

Fuente: Elaboración propia.

Caso de prueba de API			
ID	AC-7	Categoría	Control de acceso
REF	OWASP ASVS	ID REF	V4.1.8
Prerrequisitos	...		
Configuración requerida	Header con token de acceso JWT, manipulado		
Método HTTP	GET	Endpoint	api/private-scoped
Headers			
Descripción			
Intentar cambiar los atributos de un usuario para concederse permisos, cambio de permisos que no le corresponden por defecto (los permisos del usuario vienen en el token)			
Resultados esperados	401 sin autorización		
Resultado actual	401 sin autorización		
Éxito/Fallo	Éxito	Realizado/Pendiente	Realizado
Notas	Para más seguridad se debe replicar en todos los Endpoint, que requieran permisos específicos		

Para verificar el control de seguridad V4.1.8 del estándar ASVS 4.0.3, que establece: "Verificar que todos los atributos de usuario, datos y políticas utilizados por los controles de acceso no puedan ser manipulados por los usuarios finales, a menos que estén específicamente autorizados", se realizó el caso de prueba detallado en la tabla anterior. En este caso, se llevó a cabo una verificación manual de los datos que identifican al usuario, como su ID (sub) y sus permisos.

Estos datos se incluyen en el encabezado de cada solicitud legítima. Como ventaja del mecanismo de autenticación y autorización utilizado (JWT), descrito en la práctica SA-DA-2, los tokens de acceso no pueden ser modificados debido al uso de la firma RS256. Si el token es alterado, la firma se invalida y el token es considerado ilegítimo. Además, no es posible revertir un token codificado a un estado válido, lo que garantiza la integridad y autenticidad de la información.

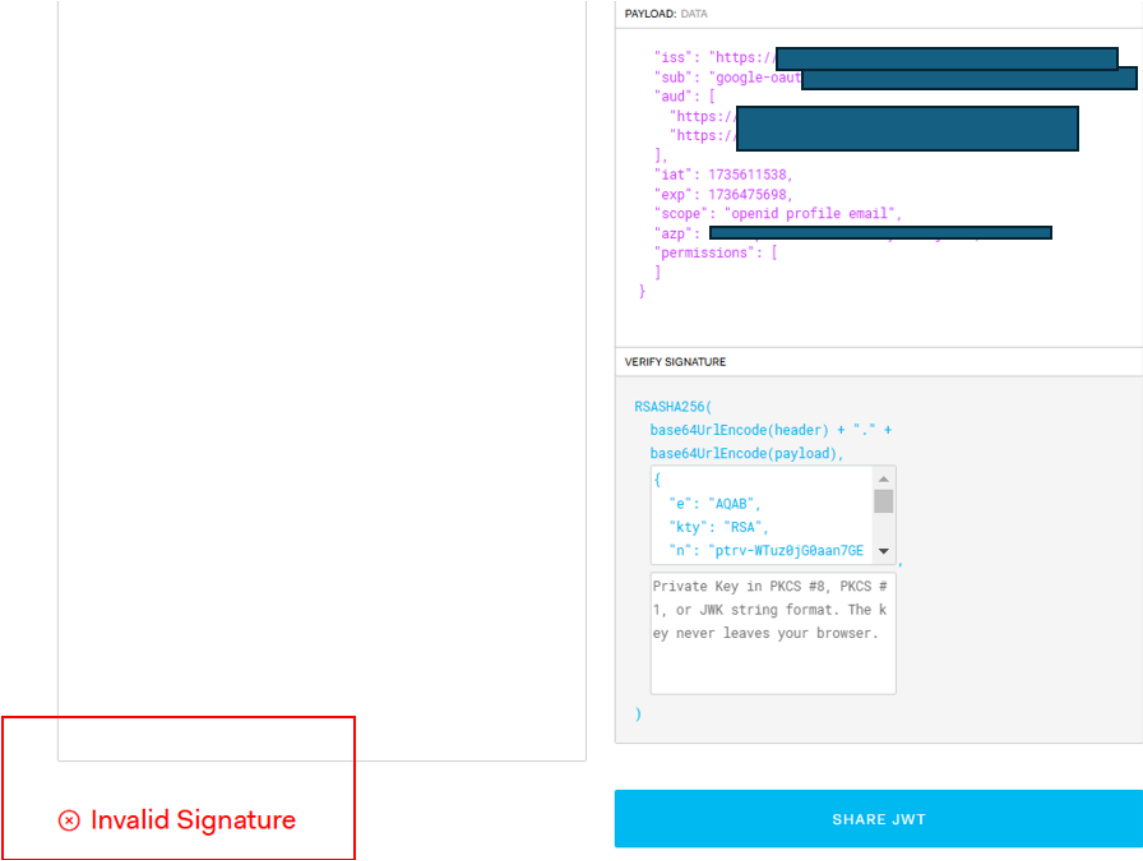


Figura 46: Validación AC-1.

Fuente: Elaboración propia.

Tabla 18: Caso de prueba AC-8.

Fuente: Elaboración propia.

Caso de prueba de API			
ID	AC-8	Categoría	Control de acceso
REF	OWASP ASVS	ID REF	V4.1.3
Prerrequisitos	...		
Configuración requerida	Header con token de acceso valido		
Método HTTP	GET	Endpoint	api/private-scoped

Headers			
Descripción	Solicitud a endpoint con un usuario sin el rol requerido (premium), no permita el acceso, pero con token valido		
Resultados esperados			
Resultado actual			
Éxito/Fallo		Realizado/Pendiente	Pendiente
Notas	Para más seguridad se debe replicar en todos los Endpoint, que requieran permisos específicos		

Tabla 19: Caso de prueba AC-9.

Fuente: Elaboración propia.

Caso de prueba de API			
ID	AC-9	Categoría	Control de acceso
REF	OWASP ASVS	ID REF	V4.2.1
Prerrequisitos	...		
Configuración requerida	Header con token de acceso valido para el usuario normal		
Método HTTP	GET	Endpoint	api/private-scoped
Headers			
Descripción	Modificar un objeto (ejemplo: food objeto), de un usuario que no le corresponde. Primero agregar a un usuario un objeto, luego con otro usuario intentar modificar ese objeto que no le pertenece		
Resultados esperados	Fallo		
Resultado actual	-		
Éxito/Fallo	-	Realizado/Pendiente	Pendiente
Notas	Para más seguridad se debe replicar en todos los Endpoint, donde se modifica un objeto (DELETE PUT)		

Validación de entradas

Tabla 20: Caso de prueba IV-1.

Fuente: Elaboración propia.

Caso de prueba de API			
ID	IV-1	Categoría	Validación de entradas
REF	OWASP ASVS	ID REF	V5.1.2

Prerrequisitos	...		
Configuración requerida	...		
Método HTTP	GET	Endpoint	/api/profile/user
Headers	Token de acceso valido		
Entradas	En el Body de la solicitud Ingresar un dato más de los permitidos en formato json		
Descripción			
Realizar una solicitud a un endpoint con un campo no permitido, campos permitidos: 'intolerance_allergies', 'home_size', 'name_user', 'age', 'month_budget', 'phone', 'biography'			
Resultados esperados	El campo no permitido, sea ignorado antes de llegar al endpoint		
Resultado actual			
Éxito/Fallo	Éxito	Realizado/Pendiente	Realizado
Notas	Para más seguridad se debe replicar en todos los Endpoint		

El control de referencia ASVS V5.1.2 establece: “Verifica que los *frameworks* protejan contra ataques de asignación masiva de parámetros, o que la aplicación tenga contramedidas para protegerse contra la asignación insegura de parámetros, como marcar campos como privados o similares”.

En el caso de Django, al utilizar serializadores, únicamente los campos definidos explícitamente en ellos serán aceptados por el framework, mientras que los campos no definidos serán ignorados. Por esta razón, Django recomienda utilizar serializadores de manera explícita, evitando el uso de la notación `fields = '__all__'`, ya que esta podría exponer campos no deseados. A continuación, se muestra un ejemplo:



```

class UserDataSerializer(serializers.ModelSerializer):
    class Meta:
        model = UserData
        fields = '__all__'

class UserDataGetData(serializers.ModelSerializer):
    class Meta:
        model = UserData
        fields = ['intolerance_allergies', 'home_size', 'name_user', 'age', 'month_budget', 'phone', 'biography']

```

Figura 47: Verificación IV-1.

Fuente: Elaboración propia.

Tabla 21: Caso de prueba IV-5.

Fuente: Elaboración propia.

Caso de prueba de API			
ID	IV-5	Categoría	Validación de entradas

REF	OWASP ASVS	ID REF	V5.1.4
Prerrequisitos			
Configuración requerida	Token de acceso valido		
Método HTTP	POST	Endpoint	api/*
Headers			
Entradas	Cuerpo con datos de validación inválidos ej: numero +569123456789		
Descripción			
Realizar una POST a un endpoint, y revisar que los datos estructurados estén fuertemente tipados y validados contra un esquema definido que incluya caracteres permitidos, longitud y patrones (por ejemplo, números de tarjetas de crédito, correos electrónicos, números telefónicos).			
Resultados esperados	400		
Resultado actual			
Éxito/Fallo	Éxito	Realizado/Pendiente	Realizado
Notas			

El estándar ASVS, en el control V5.1.4, establece: “Verificar que los datos estructurados estén fuertemente tipados y validados contra un esquema definido que incluya caracteres permitidos, longitud y patrones (por ejemplo, números de tarjetas de crédito, correos electrónicos, números telefónicos)”.

En este contexto, es fundamental verificar todas las entradas que reciban datos fuertemente tipados. En la API de Not Waste, actualmente en desarrollo, solo existe un endpoint que recibe un dato de este tipo: api/profile/user. Este endpoint permite realizar solicitudes PUT para modificar parte del perfil del usuario. El dato que debe ser validado es el campo phone, para el cual se define un conjunto de datos válidos e inválidos.

Por el momento, las validaciones implementadas solo aceptan números telefónicos chilenos, ya que la lógica de negocio está definida de acuerdo con este criterio.

```
...
El número de teléfono +56912345678 es válido en Chile.
El número de teléfono +56998765432 es válido en Chile.
El número de teléfono +56 9 1234 5678 no es válido en Chile.
El número de teléfono +1 800 123 4567 no es válido en Chile.
El número de teléfono +34 612 345 678 no es válido en Chile.
El número de teléfono 123456789 no es válido en Chile.
El número de teléfono abcd1234 no es válido en Chile.
...
```

Figura 48: Números válidos.
Fuente Elaboración propia.

En la siguiente figura se muestra una prueba manual con Postman para este caso de prueba:

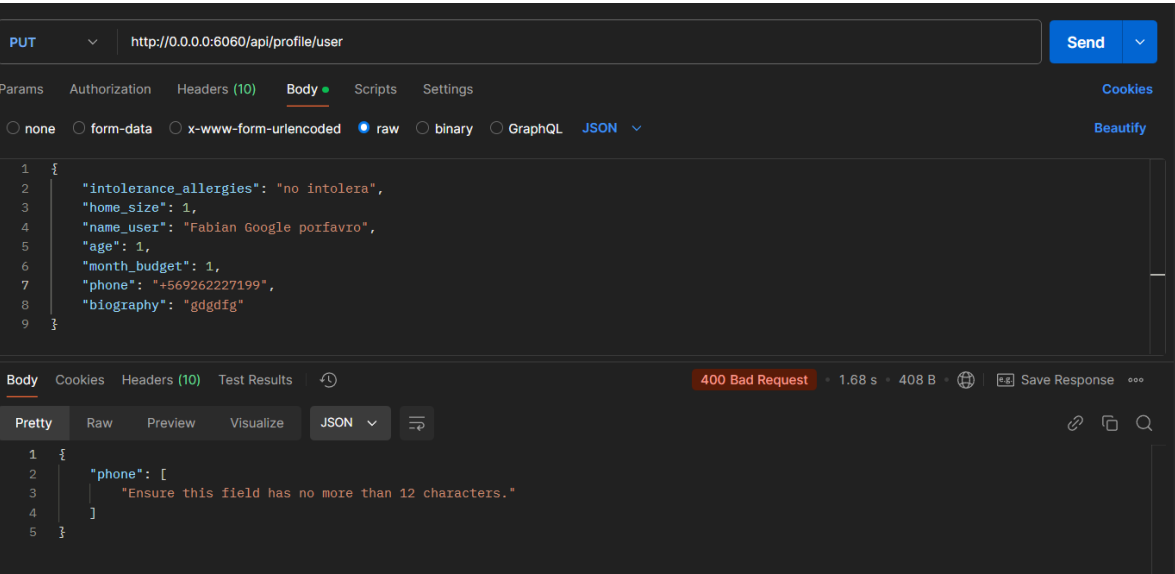


Figura 49: Prueba IV-5.
Fuente: Elaboración propia.

Tabla 22: Caso de prueba IV-8.
Fuente: Elaboración propia.

Caso de prueba de API			
ID	IV-8	Categoría	Validación de entradas
REF	OWASP ASVS	ID REF	V5.3.4
Prerrequisitos	...		
Configuración requerida	...		
Método HTTP	...	Endpoint	api/*
Headers	...		

Entradas	...		
Descripción	Revisar que se use Django ORM		
Resultados esperados	...		
Resultado actual	...		
Éxito/Fallo	Éxito	Realizado/Pendiente	Realizado
Notas	Django ORM abstraen las consultas SQL y, por defecto, utilizan consultas parametrizadas		

Este control se refiere a ASVS 4.0.3 V5.3.4, se dedica a “Verificar que las consultas de selección de datos o bases de datos (por ejemplo, SQL, HQL, ORM, NoSQL) usen consultas parametrizadas o mecanismos similares para proteger contra ataques de inyección de bases de datos” la Ventaja de Django en este contexto puede ser encontrado en su propia documentación Seguridad en Django (Django, n.d.) “Los conjuntos de consultas de Django están protegidos contra la inyección SQL, ya que sus consultas se construyen utilizando parametrización de consultas. El código SQL de una consulta se define por separado de los parámetros de la consulta. Dado que los parámetros pueden ser proporcionados por el usuario y, por lo tanto, no son seguros, el controlador de base de datos subyacente los escapa.”

Archivos y recursos

Tabla 23: Caso de prueba FR-1.

Fuente: Elaboración propia.

Caso de prueba de API			
ID	FR-1	Categoría	Validación de entradas
REF	OWASP ASVS	ID REF	V12.1.1
Prerrequisitos	...		
Configuración requerida	...		
Método HTTP	POST	Endpoint	api/food/
Headers	Token de acceso valido		
Entradas	Imagen con archivo (tipo jpg) tamaño superior 80 kb		
Descripción			
Enviar una petición con un archivo tipo jpg/png que supere el valor límite de 80 Kb, en un endpoint que lo reciba imágenes en este caso POST api/food/			
Resultados esperados	ERROR archivo superior al tamaño permitido		
Resultado actual	...		
Éxito/Fallo		Realizado/Pendiente	Pendiente
Notas	Realizar en todos los endpoint que acepten FILES image, probar con un archivo valido y uno que supere lo permitido		

Configuración

Tabla 24: Caso de prueba CF-1.

Fuente: Elaboración propia.

Caso de prueba de API			
ID	CF-1	Categoría	Configuración
REF	OWASP ASVS	ID REF	V14.4.1
Prerrequisitos	...		
Configuración requerida	...		
Método HTTP	GET	Endpoint	api/*
Headers	...		
Entradas	...		
Descripción			
Enviar una petición para verificar que cada respuesta HTTP contenga un encabezado Content-Type.			
Resultados esperados	Content-Type Json en el encabezado		
Resultado actual	Content -Type Json en el encabezado		
Éxito/Fallo	Éxito	Realizado/Pendiente	Realizado
Notas	Se debe revisar que todos los endpoint tengan este encabezado, también se puede usar un mock, para evitar enviar un token, si es que se realiza una prueba unitaria		

```

@pytest.mark.django_db
def test_json_content_type_with_authorization(client):
    # Obtener el token real (simulado aquí)
    token_response = get_auth_token()
    access_token = token_response["access_token"]

    # Define las rutas que deben responder con JSON
    endpoints = [
        reverse("food-api"), # endpoint para probar
    ]

    # Encabezados para las solicitudes
    headers = {
        "Authorization": access_token,
    }

    for endpoint in endpoints:
        response = client.get(endpoint, **{"HTTP_AUTHORIZATION": headers["Authorization"]})

        # Verificar que el encabezado Content-Type esté presente
        assert "Content-Type" in response.headers, f"Falta el encabezado Content-Type en {endpoint}"

        content_type = response.headers["Content-Type"]

        # Verificar que Content-Type sea application/json
        assert content_type == "application/json", (
            f"El encabezado Content-Type en {endpoint} debería ser 'application/json', pero es '{content_type}'."
        )

        # Verificar que el contenido sea JSON válido
        try:
            response.json() # Intenta parsear el contenido como JSON
        except ValueError:
            pytest.fail(f"El contenido de la respuesta en {endpoint} no es JSON válido.")

```

Figura 50: Prueba CF-1

Fuente: Elaboración propia.

Tabla 25: Caso de prueba CF-2.

Fuente: Elaboración propia.

Caso de prueba de API			
ID	CF-2	Categoría	Configuración
REF	OWASP ASVS	ID REF	V14.3.2
Prerrequisitos	El servidor debe estar en un entorno de preparación o producción		
Configuración requerida	...		
Método HTTP	*	Endpoint	api/*
Headers	..		
Entradas	...		
Descripción			
Hacer una petición y revisar que no exista una pantalla que muestre información de debug, revisar que en el archivo de configuración debug este en false			
Resultados esperados	No información de debug		
Resultado actual	Respuesta esperada		
Éxito/Fallo	Éxito	Realizado/Pendiente	Realizado

Notas	• Revisar que los log no guarden información de nivel debug • Eliminar logging de clase logging.StreamHandler • Ejecutar comando python manage.py check --deploy
--------------	--

El control V14.3.2 del estándar ASVS establece: “Verificar que los modos de depuración del servidor web, la aplicación y el marco de la aplicación estén desactivados en producción, para eliminar características de depuración, consolas de desarrolladores y divulgaciones no intencionadas de seguridad”.

En la revisión manual realizada, se verificaron las configuraciones y se constató que el modo de depuración está desactivado (con el valor false). Además, se confirmó que no se está utilizando un Handler de consola para los registros (logs).

```

GNU nano 7.2 settings.py
Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).resolve().parent.parent

dotenv.load_dotenv(BASE_DIR / '.env')

MEDIA_URL = '/media/'
MEDIA_ROOT = BASE_DIR / 'media'

Quick-start development settings - unsuitable for production
See https://docs.djangoproject.com/en/5.0/howto/deployment/checklist/

SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = get_env_var('DJANGO_SECRET_KEY')

SECURITY WARNING: don't run with debug turned on in production!
DEBUG = False

ALLOWED_HOSTS = ['api.notwaste.tech']
CORS_ALLOW_ALL_ORIGINS = True

SECURE_SSL_REDIRECT = True
SECURE_PROXY_SSL_HEADER = ('HTTP_X_FORWARDED_PROTO', 'https')
SESSION_COOKIE_SECURE = True
SESSION_COOKIE_SAMESITE = None
SECURE_HSTS_SECONDS = 15724800
SECURE_HSTS_INCLUDE_SUBDOMAINS = True
SECURE_HSTS_PRELOAD = True

```

Figura 51: Verificación CF-2.

Fuente: Elaboración propia.

Tabla 26: Caso de prueba CF-3.

Fuente: Elaboración propia.

Caso de prueba de API			
ID	CF-3	Categoría	Configuración
REF	OWASP ASVS	ID REF	V14.3.3
Prerrequisitos	...		
Configuración requerida	...		
Método HTTP	GET	Endpoint	api/*
Headers	..		
Entradas	...		
Descripción			

Realizar solicitudes y revisar que no expongan información detallada sobre las versiones de los componentes del sistema. (powered-by)			
Resultados esperados	Respuesta sin información detalla del servidor, ejemplo: versión Nginx, Gunicorn, etc.		
Resultado actual	Respuesta esperada		
Éxito/Fallo	Éxito	Realizado/Pendiente	Realizado
Notas	Replicar prueba en todos los endpoint		

El control referenciado es el siguiente del ASVS V14.3.3 “Verificar que los encabezados HTTP o cualquier parte de la respuesta HTTP no expongan información detallada sobre las versiones de los componentes del sistema.”

En este caso se realizaron dos pruebas manuales, una se realizó enviando una petición directamente a IP del servidor, y otra enviando la petición al dominio del servidor a continuación la petición al dominio y posteriormente a la IP del servidor.

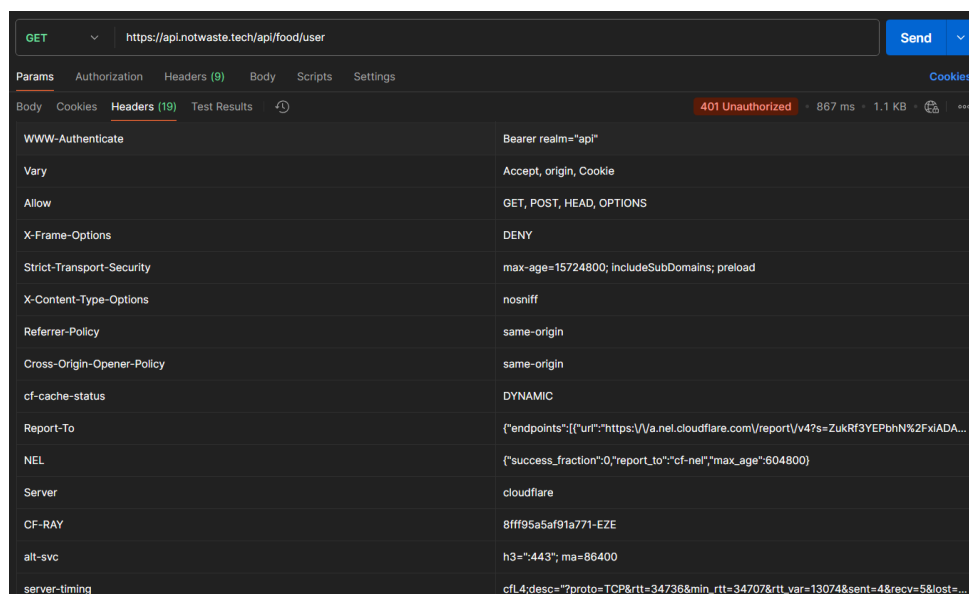


Figura 52: Prueba CF-3 parte 1.

Fuente: Elaboración propia.

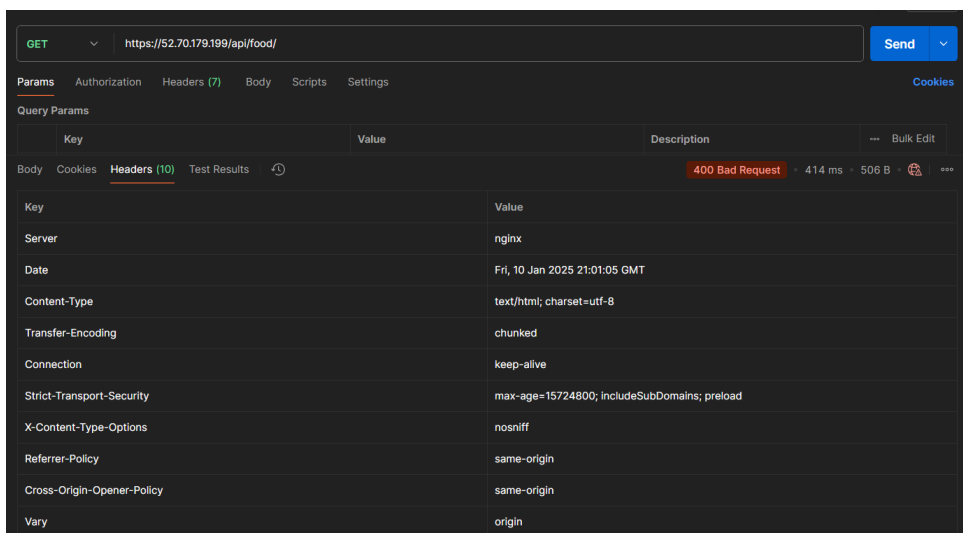


Figura 53: Prueba CF-3 parte 2.
Fuente: Elaboración propia.

Tabla 27: Caso de prueba CF-4.

Fuente: Elaboración propia.

Caso de prueba de API			
ID	CF-4	Categoría	Configuración
REF	OWASP ASVS	ID REF	V14.4.4
Prerrequisitos	...		
Configuración requerida	...		
Método HTTP	GET	Endpoint	api/*
Headers	..		
Entradas	...		
Descripción			
Realizar peticiones GET y verificar que todas las respuestas contengan un encabezado X-Content-Type-Options: nosniff.			
Resultados esperados	Respuesta con Header Content-Type-Options: nosniff.		
Resultado actual	Respuesta esperada		
Éxito/Fallo	Éxito	Realizado/Pendiente	Realizado
Notas	Replicar prueba en todos los endpoint		

En control del ASVS V14.4.4 consiste en “Verificar que todas las respuestas contengan un encabezado X-Content-Type-Options: nosniff”, la respuesta de la petición realizada muestra la respuesta deseada.

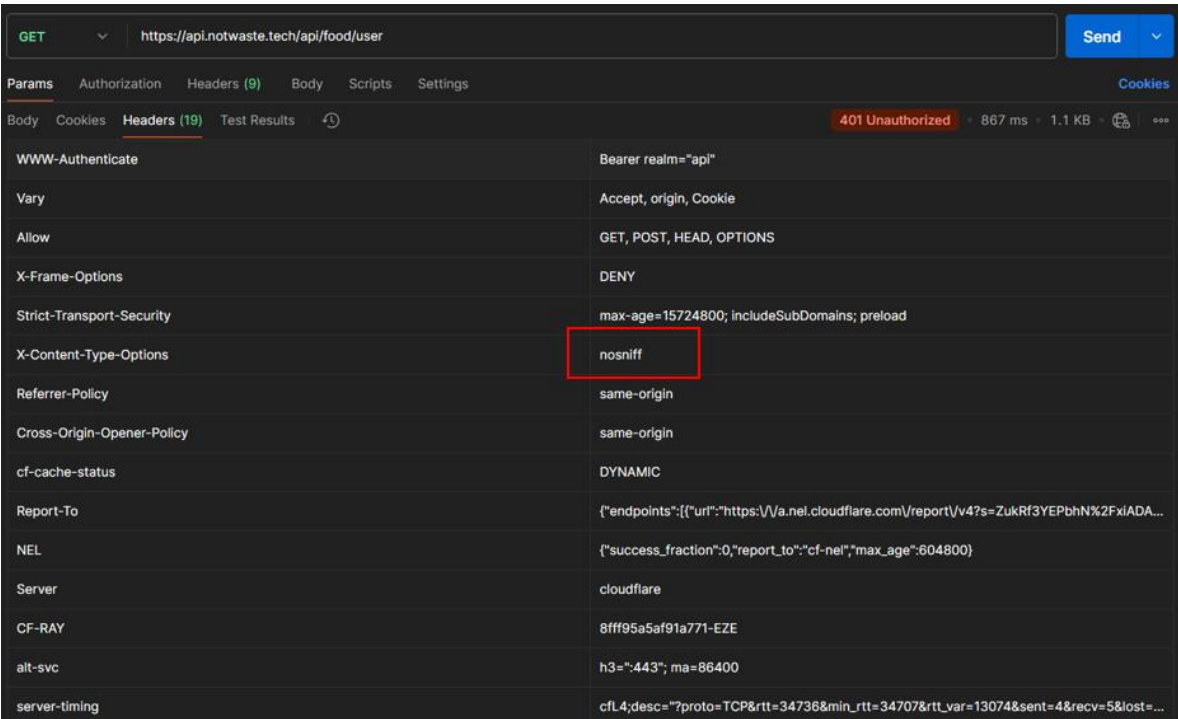


Figura 54:Prueba CF-4.

Fuente: Elaboración propia.

Tabla 28: Caso de prueba CF-5.

Fuente: Elaboración propia.

Caso de prueba de API			
ID	CF-5	Categoría	Configuración
REF	OWASP ASVS	ID REF	V14.4.5
Prerrequisitos	...		
Configuración requerida	...		
Método HTTP	GET	Endpoint	api/*
Headers	..		
Entradas	...		
Descripción			
Enviar petición y revisar si el Header de la respuesta tiene: Strict-Transport-Security esté incluido en todas las respuestas y para todos los subdominios, como Strict-Transport-Security: max-age=15724800; includeSubdomains.			
Resultados esperados	Strict-Transport-Security max-age=15724800; includeSubDomains; preload		
Resultado actual			
Éxito/Fallo		Realizado/Pendiente	
Notas	En Django el comando verifica HSTS, HSTS_SECONDS, HSTS include subdomain: python manage.py check --deploy		

El control del ASVS V14.4.5 es “Verificar que un encabezado Strict-Transport-Security esté incluido en todas las respuestas y para todos los subdominios, como Strict-Transport-Security: max-age=15724800; includeSubdomains.” Al revisar los encabezados en la petición son correctos.

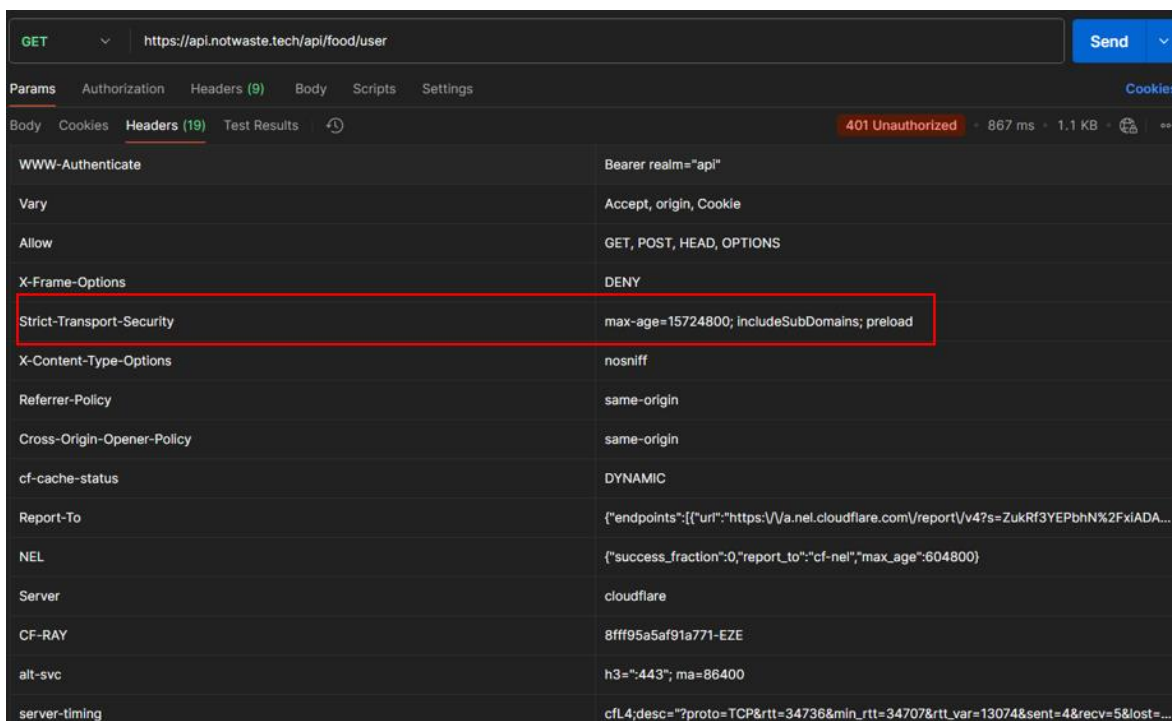


Figura 55: Prueba CF-5, Postman.

Fuente: Elaboración propia.

4.3.6 Resumen de práctica

La realización de pruebas de seguridad fue fundamental para identificar vulnerabilidades y determinar las áreas en las que el software necesita mejoras, así como para fomentar el uso de buenas prácticas.

Aunque se ha implementado un conjunto limitado de controles de seguridad, aún quedan muchos controles por evaluar. Sin embargo, incluso con pocas pruebas realizadas, se han identificado mejoras importantes que no se habían considerado inicialmente, como una correcta configuración y una adecuada validación de entradas. Además, se ha verificado que ciertos controles de seguridad, como el control de acceso, funcionan según lo esperado.

Es importante utilizar una guía de referencia antes de iniciar el desarrollo. El uso de guías reconocidas, como las propuestas por organizaciones como OWASP, es crucial para identificar controles de seguridad relevantes que ya han sido estudiados y probados.

4.4 Resultados

Se llevará a cabo un análisis de los criterios de calidad necesarios para considerar completado el nivel de madurez de la práctica de seguridad implementada. En este análisis se presentará una tabla que especificará los criterios de calidad asociados a la práctica, junto con una evaluación sobre si estos criterios fueron cumplidos.

La tabla detalla las preguntas que identifican las acciones sugeridas por la práctica de seguridad, así como los criterios de calidad que determinan su cumplimiento. En la sección de observaciones se proporciona un contexto adicional sobre los criterios de calidad, incluyendo el progreso alcanzado en la implementación de la práctica y los aspectos pendientes para lograr su cumplimiento. Para obtener un contexto más detallado sobre las preguntas y los criterios de calidad, se sugiere consultar el capítulo 3, donde se explica la estructura de SAMM y se proporciona una descripción exhaustiva de estos elementos.

De acuerdo con las directrices de SAMM, para alcanzar el nivel de madurez completo de una práctica de seguridad, es imprescindible que todos los criterios de calidad de la actividad estén cumplidos. Si alguno de los criterios no se cumple, la evaluación no puede superar un puntaje de 0. En el caso de Not Waste, considerando el alcance definido inicialmente (API Django), los criterios de calidad no se alcanzaron en su totalidad, es decir que el puntaje después de la realización de las actividades hasta el momento sigue en 0 para Not Waste.

A pesar de que no se logró completar todos los criterios de calidad, se evidenció un avance significativo en la implementación general de la práctica de seguridad, cumpliendo parcialmente con sus objetivos.

4.4.1 Resultados práctica SA-AD-2

Tabla 29: Evaluación práctica SA-AD-2.

Fuente: Elaboración propia.

¿Utiliza servicios de seguridad compartidos durante el diseño? (nivel de madurez 2)		
Criterios de calidad	Respuesta	Observaciones

Tiene una lista documentada de servicios de seguridad reutilizables, disponible para las partes interesadas relevantes	No	Existe una lista de servicios, sin embargo, falta una estructura de documentación, más amplia para los servicios utilizados y para los posibles servicios que podrían ser utilizados.
Ha revisado la postura de seguridad de referencia para cada servicio seleccionado	No	Se ha realizado una revisión de los conceptos de seguridad para los servicios, pero falta profundidad, ej: ABAC, Registros provenientes de otros componentes del sistema.
Sus diseñadores están capacitados para integrar cada servicio seleccionado siguiendo las pautas disponibles.	No	Not Waste carece de diseñadores capacitados en los servicios aplicados.

Mejoras conseguidas

Se han implementado servicios de seguridad como Auth0, Cloudflare y Better Stack, logrando mejoras significativas gracias a la aplicación de la práctica de seguridad SA-AD-2. Para profundizar en las mejoras aportadas por estos servicios a Not Waste, en cada sección de implementación de la práctica SA-AD-2 se incluyeron ejemplos específicos que destacan las funcionalidades clave de estos servicios. A continuación, se presentan los principales beneficios que cada uno de estos servicios ha proporcionado:

- **Auth0:** Este servicio facilita la autenticación y autorización de la API mediante el uso de tokens, eliminando la necesidad de que la API maneje credenciales directamente. Esto permite segmentar las responsabilidades entre la lógica de negocio y los procesos de autenticación y autorización, aumentando la seguridad y simplificando la gestión de accesos.
- **Cloudflare:** Se utilizó su firewall de aplicaciones (WAF) para restringir el acceso a la API únicamente a usuarios autorizados. Esto previene que el tráfico no autorizado inunde la red y consuma recursos innecesarios.

- **Better Stack:** Este servicio permitió centralizar y analizar registros provenientes de diversas fuentes en un único lugar, estructurándolos para facilitar su revisión. Además, se habilitó la generación de métricas personalizadas, lo que ayudó a realizar análisis periódicos de seguridad. Gracias a Better Stack, se pudieron tomar acciones como el bloqueo de direcciones IP maliciosas y la mejora de configuraciones para garantizar que solo las conexiones que pasaran por el WAF fueran aceptadas.

4.4.2 Resultados práctica RT-CV-1

Tabla 30: Evaluación práctica RT-CV-1.

Fuente: Elaboración propia.

¿Prueba las aplicaciones para comprobar el correcto funcionamiento de los controles de seguridad estándar? (Nivel de madurez 1)		
Criterios de calidad	Respuesta	Observaciones
Las pruebas de seguridad al menos verifican la implementación de autenticación, control de acceso, validación de entrada, codificación y escape de datos y controles de cifrado.	No	• Se realizaron pruebas de control de acceso y validación de entrada, y se realizaron otros tipos de pruebas como comunicación, configuración, manejo de errores. • Faltaron pruebas de autenticación, escape de datos y controles de cifrado. • Esta práctica de seguridad comprende la realización de pruebas a pruebas manuales o las pruebas posibles de desarrollo de acuerdo al contexto de las organizaciones, pero se hicieron pruebas automáticas y manuales.

Las pruebas de seguridad se ejecutan siempre que la aplicación cambia su uso de los controles.	No	No aplicable por el momento.
--	----	------------------------------

Mejoras conseguidas

Las mejoras en la implementación de la verificación de controles de seguridad consistieron en identificar si los controles de seguridad estaban implementados de acuerdo con un estándar de verificación de aplicaciones web, específicamente el estándar ASVS 4.0.3. En el caso de *Not Waste*, dado que la API se encuentra en etapa de desarrollo, se observó que ciertos controles no habían sido aplicados. Esto proporcionó un contexto más claro sobre las consideraciones de seguridad que deben tenerse en cuenta para la API.

Entre los aspectos evaluados que fueron mejorados, se destacaron áreas como la configuración adecuada según el estándar, la validación de entradas y la implementación de mecanismos seguros de control de acceso. Además, esta evaluación ofreció una visión a futuro sobre los controles de seguridad adicionales que deben aplicarse, tanto para mejorar la seguridad de la API conforme al estándar ASVS, como para fortalecer la seguridad de la aplicación móvil utilizando MASVS.

CAPÍTULO 5: CONCLUSIONES

En este capítulo se analizará cómo se logró cumplir con el objetivo general de este trabajo: implementar prácticas de seguridad en Not Waste. Para ello, se utilizó OWASP SAMM como metodología para identificar las prácticas de seguridad necesarias y se aplicó su estrategia de implementación en el caso de estudio de Not Waste.

Asimismo, se abordarán las limitaciones del marco de seguridad aplicado (SAMM), particularmente en relación con la falta de información detallada sobre la ejecución de ciertas actividades y cómo realizarlas con calidad, lo que puede representar un desafío para organizaciones pequeñas.

Se incluirán recomendaciones sobre la importancia de comprender y planificar adecuadamente antes de implementar el marco de seguridad, junto con herramientas recomendadas. También se comparará el uso de SAMM frente a otras alternativas, como BSIMM o ISO 27001, destacando su idoneidad para startups de desarrollo de software.

Finalmente, se discutirán posibles líneas de trabajo futuro, enfocadas en mejorar tanto las prácticas de seguridad implementadas como el marco de seguridad en general dentro del contexto de Not Waste.

5.1 Cumplimiento Objetivo General

Objetivo General: Implementar prácticas de seguridad informática utilizando una estrategia basada en el modelo de madurez OWASP SAMM y su metodología de aplicación, con el propósito de reforzar la seguridad de la aplicación Not Waste y proteger sus activos frente a posibles amenazas.

Tal como se presentó en el capítulo 4, correspondiente a la validación de la solución, se implementaron prácticas de seguridad basadas en OWASP SAMM. Por un lado, se adoptó una práctica de seguridad relacionada con **Architecture Design**, en particular: “Establecer patrones de diseño comunes y soluciones de seguridad para su adopción”. Por otro lado, se llevó a cabo una práctica de **Requirements-driven Testing**, que consiste en la prueba de controles de seguridad del software.

Aunque las prácticas implementadas no alcanzaron los criterios de calidad necesarios para situarlas en el nivel de madurez correspondiente, ambas contribuyeron con mejoras significativas que fueron detalladas en los capítulos dedicados a su implementación.

La práctica de **Architecture Design** implicó mejoras en la implementación de servicios de seguridad, generando beneficios concretos como:

- Una autenticación más segura.
- Mecanismos para evitar tráfico no autorizado al servidor.
- Mayor visibilidad del tráfico en el servidor, lo que derivó en medidas específicas de configuración para aumentar la seguridad de la aplicación Not Waste.

En cuanto a la práctica de **Requirements-driven Testing**, se logró:

- Mejorar las prácticas de control de seguridad que previamente no habían sido implementadas.
- Establecer un procedimiento para probar controles de seguridad.
- Verificar controles de seguridad en la aplicación, fortaleciendo su protección frente a posibles vulnerabilidades.

Aunque las prácticas no alcanzaron el nivel de madurez esperado, su implementación marcó un avance significativo en la seguridad de *Not Waste*, estableciendo una base sólida para futuras mejoras que se detallarán más adelante.

Finalmente, se alcanzó el objetivo de implementar prácticas de seguridad en la aplicación, aplicando las directrices descritas a través de SAMM, lo que reforzó la seguridad de *Not Waste*.

5.2 Limitaciones del Modelo aplicado

El modelo OWASP SAMM presenta algunas limitaciones que se evidenciaron en la aplicación del modelo en Not Waste y que deben ser abordadas por las organizaciones para garantizar la implementación de actividades con calidad en todos los procesos llevados a cabo.

Una de estas limitaciones radica en que las actividades de seguridad propuestas por SAMM no exploran en profundidad las buenas prácticas necesarias para su correcta ejecución. Por ejemplo, en la actividad de "Verificación de control" que se implementó en Not Waste "prueba de controles de seguridad", el modelo no detalla cómo realizar pruebas de calidad ni qué metodologías específicas emplear.

Asimismo, los pasos que explican la aplicación del modelo, tales como preparación, definición de objetivos, planificación e implementación, presentan vacíos en términos de recursos prácticos. Por ejemplo, en la etapa de planificación se sugiere elaborar un cronograma para determinar cuándo realizar las tareas, pero no se ofrecen guías ni herramientas concretas que permitan a las organizaciones realizar una estimación precisa de tiempo, dejando esta responsabilidad completamente al criterio de quien lo desarrolla.

Por un lado, esta flexibilidad permite que cada organización adapte los procesos según sus propias necesidades y contextos. Sin embargo, también representa un desafío significativo para organizaciones emergentes, como Not Waste, que están en proceso de implementar el marco de seguridad y carecen de un conocimiento profundo sobre ciertos conceptos clave. Si bien no es el propósito principal de SAMM abordar todos estos aspectos, la falta de recursos detallados puede dificultar la adopción eficiente del modelo en las organizaciones.

Es importante destacar que SAMM incluye referencias a otros marcos y guías para facilitar la implementación de prácticas específicas. Sin embargo, esta ventaja no se extiende a los pasos de aplicación del modelo, y una ayuda seria incorporar recomendaciones de estándares como el PMBOK permitiría a las organizaciones realizar cronogramas más precisos y mejorar la planificación general de sus actividades.

5.3 Recomendaciones

Se presentarán recomendaciones clave basadas en el aprendizaje obtenido de la investigación de marcos de seguridad como BSIMM, ISO 27001 y la aplicación de SAMM como base para implementar prácticas de seguridad en el caso de estudio *Not Waste*. Estas recomendaciones abarcan la importancia de realizar pasos previos esenciales para garantizar una implementación efectiva, así como la identificación de herramientas recomendadas utilizadas en la práctica de seguridad SA-DA-2. Finalmente, se ofrecerá una recomendación para las startups de desarrollo de software basada en la investigación de los marcos de seguridad BSIMM, SAMM e ISO 27001, considerando el aprendizaje obtenido durante el proceso.

5.3.1 Importancia de pasos previos

Antes de iniciar la exploración e implementación de prácticas de seguridad, resulta esencial comprender a fondo los pasos previos a la ejecución. En el contexto de la aplicación de las prácticas de seguridad de OWASP SAMM para el caso de estudio *Not Waste*, se comprendió la importancia fundamental de estas etapas iniciales, que incluyen la preparación, la evaluación, la definición del plan y el establecimiento de objetivos.

Durante los primeros intentos de aplicación, la falta de conocimiento sobre estos pasos previos generó confusión y frustración, ya que no se contaba con una dirección clara sobre cómo proceder. Esto dio lugar a preguntas como: ¿qué camino de implementación seguir? ¿qué prácticas priorizar? La ausencia de una estructura definida provocó desorganización, dejando algunas tareas sin completar y, por consecuencia, fuera de este informe.

Esta experiencia refleja un desafío al que podrían enfrentarse algunas organizaciones al intentar implementar SAMM como marco de seguridad: la incapacidad de priorizar adecuadamente las actividades y prácticas puede hacer que los esfuerzos hacia la mejora de la seguridad se vuelvan confusos y, en algunos casos, ineficaces. Por ello, asegurar una adecuada planificación inicial y entender por completo los pasos de aplicación son claves para garantizar un progreso ordenado y efectivo en la implementación de prácticas de seguridad que ofrece este modelo de madurez.

Complementación con otras metodologías

OWASP SAMM proporciona dos tipos de guías para la realización de actividades: las guías de la comunidad y las guías desarrolladas por el equipo de SAMM. Sin embargo, estas presentan diferencias significativas en cuanto a su disponibilidad y alcance. Las guías de la comunidad no están presentes en todas las actividades, mientras que las guías del equipo de SAMM sí lo están, aunque no de manera completamente exhaustiva. Ambas ofrecen

enlaces de apoyo para aspectos específicos de las actividades, pero no abarcan todos los elementos involucrados en las actividades.

No obstante, en las guías del equipo de SAMM se incluye un enlace a OPENCRE, una herramienta valiosa para mapear actividades con otros marcos de seguridad. Por ejemplo, en el flujo A de la fase de verificación, correspondiente a la actividad "Verificación de Control", se enlaza con el código CRE: 433-442 que están vinculados a otras fuentes, tales como controles de ISO 27001:8.25 o el estándar ASVS V.1.1.1.

Estos mapeos resultan particularmente útiles, ya que permiten explorar las actividades con mayor profundidad y mejorar la calidad de las prácticas implementadas. Asimismo, ofrecen una comprensión más detallada de las actividades al integrar información proporcionada por otros marcos y estándares, enriqueciendo el proceso de implementación y aseguramiento de la seguridad, cuando se enfrente a actividades que carecen de detalles, resulta muy útil utilizar mapeos hacia actividades de otros marcos de seguridad para obtener más información.

Exploración de OWASP: Una recomendación adicional consiste en la exploración de otros recursos proporcionados por OWASP, como ASVS, MASVS, OWASP Cheat Sheets, OWASP Top 10 y OWASP API Security Top 10. Estos recursos se destacan por ofrecer guías rápidas y altamente compatibles con OWASP SAMM, facilitando la implementación de diversas actividades. Por ejemplo, el estándar ASVS fue empleado en la implementación de SAMM en Not Waste, para establecer controles de prueba, mientras que OWASP API Security Top 10 permitió identificar y analizar vulnerabilidades específicas en la API que constituía el principal objetivo de protección. En este contexto, la integración de estos recursos se presenta como una estrategia para apoyar y fortalecer la implementación de las actividades definidas en el marco SAMM de manera sencilla.

5.3.2 Herramientas recomendadas

Se utilizaron varias herramientas a lo largo de este trabajo, las cuales fueron seleccionadas principalmente por sus capas gratuitas, su documentación detallada y la facilidad de integración con las tecnologías empleadas. Pytest resultó ser muy útil para realizar pruebas en Django, ya que cuenta con una gran comunidad que proporciona información valiosa sobre su uso. Por otro lado, Postman facilitó enormemente la revisión de las respuestas de la API, lo que optimizó el proceso de testeo. No obstante, es necesario hacer un especial énfasis en dos herramientas utilizadas, que fueron muy útiles en el contexto de las prácticas de seguridad de este trabajo, Cloudflare y Auth0.

Cloudflare, como WAF (Firewall de aplicaciones), es altamente recomendable debido a varias razones. En primer lugar, ofrece reglas preconfiguradas que permiten mitigar ataques, como los provenientes de bots, tal como se evidenció anteriormente. Además, su servicio es fácil de usar y proporciona una interfaz intuitiva con diversas métricas que fueron exploradas. La integración de Cloudflare con el servidor Linux utilizado en Not Waste fue rápida, ya que solo fue necesario configurar sus Nameservers y ajustar el servidor web para utilizar el DNS. Cabe destacar que, si no se cuenta con configuración HTTPS en el servidor desde el inicio, Cloudflare proporciona HTTPS entre el cliente y Cloudflare, asegurando la comunicación.

Como se mencionó previamente en la práctica implementada SA-DA-1, se recomienda encarecidamente el uso de Auth0, ya que es un servicio que garantiza seguridad en procesos críticos como la autenticación y autorización. Este servicio es utilizado por grandes organizaciones como Pfizer, Mazda, Mercado Libre, AMD, Capcom, entre otras muchas, lo que respalda su confiabilidad y eficacia.

Auth0 ofrece una documentación extensa y de alta calidad, su documentación explica cómo implementar e integrar el servicio con una amplia variedad de tecnologías y frameworks, como Python, JavaScript, Java, Kotlin, Dart, entre otros. Aunque soluciones como Amazon Cognito también cuentan con buena documentación, Auth0 se distingue por ofrecer guías específicas para distintos casos de uso, lo que facilita su implementación. Además, en su repositorio de GitHub, proporciona una variedad de ejemplos de implementación en los frameworks y tecnologías mencionadas, y en muchos otros.

Estos recursos fueron especialmente útiles al implementarlo en la API de Django y en la aplicación móvil de Not Waste, lo que permitió reducir significativamente el tiempo y esfuerzo necesario para aplicar un flujo de autenticación seguro con Auth0.

5.3.3 Recomendación para startups

Basado en mi experiencia investigando cómo aplicar prácticas de seguridad con ISO 27001 y BSIMM, recomiendo encarecidamente la implementación de OWASP SAMM para startups dedicadas al desarrollo de productos de software. Esta metodología es especialmente útil para empresas con recursos limitados, ya que no requiere pagar por auditorías externas para evaluar las prácticas de seguridad de las aplicaciones.

Considero que SAMM es ideal para empresas que carecen de profesionales especializados en seguridad informática o que cuentan con equipos con poca experiencia en esta área, como fue el caso de mi equipo de desarrollo en Not Waste. Ya que al proporcionar una metodología de aplicación estructurada en cuatro pasos (preparación, evaluación, plan y establecimiento de objetivos) previos a la ejecución de las prácticas de seguridad fue más sencillo aplicar las practicas.

Además, al ser un modelo prescriptivo, SAMM se diferencia de BSIMM, que únicamente describe qué actividades realizar. SAMM no solo detalla qué hacer, sino también cómo hacerlo, lo que facilitó la comprensión de algunas actividades. Aunque no fue completamente claro en todos los casos, proporcionó la orientación necesaria para aplicar las actividades en Not Waste.

Por otro lado, intentar implementar BSIMM, que fue lo que se intentó en un inicio, resultó más desafiante, ya que se trata de un marco cerrado que depende de la organización que lo desarrolló (Black Duck) para llevar a cabo auditorías y evaluaciones. En contraste, SAMM ofrece su información de manera más abierta, permitiendo realizar autoevaluaciones, lo que resulta muy útil para comprender los desafíos que enfrenta una organización en cuanto a sus prácticas de seguridad.

SAMM se presenta como una herramienta invaluable al ofrecer un enfoque estructurado que permite comenzar desde cero y avanzar progresivamente hacia el desarrollo de proyectos de software protegidos contra amenazas y riesgos. Gracias a las prácticas de seguridad que establece y su enfoque escalonado se facilitó la implementación de actividades de niveles más básicos en este trabajo, sin la necesidad de iniciar con tareas complicadas y rigurosas.

5.4 Trabajo Futuro

5.4.1 Mejoras en la calidad

Se sugiere realizar una revisión más rigurosa de los servicios de seguridad utilizados, ya que solo se emplearon aquellos que contaban con una capa gratuita y que facilitaban la integración con herramientas como Auth0 o BetterStack. Además, se recomienda complementar las actividades con el uso de metodologías DevSecOps.

En cuanto al diseño, es necesario llevar a cabo una revisión más exhaustiva de las tecnologías implementadas, ya que no se exploraron completamente todas las funcionalidades que ofrecían los servicios utilizados. Por ejemplo:

- **Cloudflare:** Aún quedan posibles reglas personalizadas por implementar, como las de limitación de tasa (rate-limit).
- **Auth0:** Se deben explorar y aplicar controles adicionales como RBAC (Control de Acceso Basado en Roles) y ABAC (Control de Acceso Basado en Atributos).
- **BetterStack:** Sería útil crear métricas personalizadas y reglas que generen alertas enviadas a un canal de comunicación seguro, como Slack. También se debe implementar un mayor registro de eventos, como los de Nginx, y centralizar los registros de Cloudflare.

En la práctica de verificación implementada, se recomienda revisar más controles de seguridad y crear más casos de prueba para automatizar las pruebas que actualmente se realizan utilizando **Pytest**. Además, es fundamental verificar la aplicación móvil con el estándar **MASVS** para asegurar que cumpla con las mejores prácticas de seguridad.

5.5 Conclusión

La implementación de ciberseguridad en una startup no tiene por qué ser un proceso abrumador ni inmediato. Adoptar un enfoque gradual y metodológico, como el propuesto por OWASP SAMM, permite avanzar de manera estructurada, estableciendo objetivos claros y alcanzables. A diferencia de otros enfoques más robustos pero complejos de implementar en entornos emergentes, esta metodología se adapta a las limitaciones típicas de una startup, como la falta de recursos o de profesionales especializados en ciberseguridad.

OWASP SAMM facilita la integración de la seguridad sin requerir grandes inversiones en auditorías, al tiempo que involucra a los desarrolladores en el proceso. Esto no solo reduce la dependencia de expertos externos, sino que también promueve un aprendizaje gradual.

REFERENCIAS BIBLIOGRÁFICAS

- Auth0. (12 de 2024). *Autenticación frente a autorización*. Obtenido de Auth0:
<https://auth0.com/es/intro-to-iam/authentication-vs-authorization>
- Auth0. (12 de 2024). *Introducción a Auth0*. Obtenido de Auth0 Docs:
<https://auth0.com/docs/get-started/identity-fundamentals/introduction-to-auth0>
- Auth0. (2024). *Mobile Applications with API*. Obtenido de Auth0 by Okta:
<https://auth0.com/docs/get-started/architecture-scenarios/mobile-api>
- Auth0. (2024). *Node.js API Implementation (Mobile Apps + API)*. Obtenido de Auth0:
<https://auth0.com/docs/get-started/architecture-scenarios/mobile-api/api-implementation-nodejs>
- AWS. (2024). *¿Qué es el ciclo de vida del desarrollo de software (SDLC)?* Obtenido de AWS:
<https://aws.amazon.com/es/what-is/sdlc/>
- AWS. (2024). *Qué es la ciberseguridad*. Obtenido de AWS:
<https://aws.amazon.com/es/what-is/cybersecurity/>
- Balmaceda, L. (2024). *Django API: Authorization*. Obtenido de Auth0 :
<https://auth0.com/docs/quickstart/backend/django/01-authorization>
- Better Stack. (2024). *Introducing Better Stack: A unified platform to Spot, Resolve and Prevent downtime*. Obtenido de Better Stack:
<https://betterstack.com/press/introducing-better-stack/>
- Black Duck. (2023). *BSIMM 14 Report*.
- Black Duck. (2024). *What is DevSecOps?* Obtenido de Black Duck:
<https://www.blackduck.com/glossary/what-is-devsecops.html>
- Black Duck. (s.f.). *Seguridad de la aplicación*. Recuperado el Diciembre de 2024, de Black Duck: <https://www.blackduck.com/glossary/what-is-application-security.html>
- Carnegie Mellon University. (2010). *Spanish Language Translation of CMMI for Development, v1.3*.
- Chapple, M., & Seidl, D. (2020). *CompTIA Security+ Study Guide SY0-601, Eighth Edition*.

- Cloudflare. (2024). *¿Qué es un WAF? | Explicación de Web Application Firewall*. Obtenido de Cloudflare: <https://www.cloudflare.com/es-es/learning/ddos/glossary/web-application-firewall-waf/>
- Cloudflare. (2024). *Protección contra ataques DDoS HTTP*. Obtenido de Cloudflare: <https://developers.cloudflare.com/ddos-protection/managed-rulesets/http/>
- FAO. (29 de 09 de 2022). *Hacer frente a la pérdida y el desperdicio de alimentos: una oportunidad de ganar por partida triple*. Obtenido de FAO: <https://www.fao.org/newsroom/detail/FAO-UNEP-agriculture-environment-food-loss-waste-day-2022/es>
- FAO. (2024). *Plataforma técnica sobre la medición y la reducción de las pérdidas y el desperdicio de alimentos*. Obtenido de FAO: <https://www.fao.org/platform-food-loss-waste/background/es>
- FAO. (s.f.). *Pérdida y desperdicio de alimentos*. Recuperado el 2024, de ODEPA: <https://www.odepa.gob.cl/sustentabilidad/agricultura-sustentable/perdida-y-desperdicio-de-alimentos#:~:text=Seg%C3%BAn%20informaci%C3%B3n%20de%20la%20Organizaci%C3%B3n,toneladas%20de%20alimentos%20al%20a%C3%B1o.>
- Fortinet. (2022). *2022 Cybersecurity Skills Gap, Global Research Report*.
- Globant. (2024). *Ciclo de Vida de Desarrollo de Software (SDLC)*. Obtenido de Globant: <https://www.globant.com/es/tech-terms/ciclo-vida-desarrollo-software>
- Hovsepyan, A. (12 de 2024). *Comparing NIST SSDF and OWASP SAMM: A Comprehensive Analysis*. Obtenido de Codific: <https://codific.com/comparing-nist-ssdf-and-owasp-samm/>
- IBM. (26 de julio de 2024). *¿Qué es la seguridad de la información (InfoSec)?* Obtenido de IBM: <https://www.ibm.com/mx-es/topics/information-security>
- IBM. (2024). *Informe Cost of Data Breach*.
- Kidd, C. (Octubre de 2024). *Marcos de ciberseguridad: qué son y cómo utilizarlos*. Obtenido de Splunk.
- Maggi e Ipsos. (2023). *Primer Estudio Nacional Maggi - Ipsos sobre Desperdicio de alimentos en Chile*.
- Microsoft. (2011). *Informe de progreso del proceso SDL*. Microsoft Corporation.

- NIST. (Septiembre de 2020). *Security and Privacy Controls for Information Systems and Organizations*. Recuperado el Diciembre de 2024, de NIST, COMPUTE SECURITY RESOURCE CENTER,: <https://csrc.nist.gov/pubs/sp/800/53/r5/upd1/final>
- NIST. (11 de 2022). *About Nist*. Recuperado el 12 de 2024, de NIST: <https://www.nist.gov/about-nist>
- NIST. (2024). *The NIST Cybersecurity Framework (CSF) 2.0*.
- NIST, Kent, K., & Souppaya , M. (2006). *Guide to Computer Security Log Management, Special Publication 800-92*.
- NQA. (2013). *Guía de implementación de sistemas de gestión de seguridad de la información*. nqa. Recuperado el Diciembre de 2024
- ODEPA. (s.f.). *Pérdida y desperdicio de alimentos*. Recuperado el 2024, de ODAPA, Oficina de Estudios y Políticas Agrarias de Chile.: <https://www.odepa.gob.cl/sustentabilidad/agricultura-sustentable/perdida-y-desperdicio-de-alimentos>
- ONU. (27 de Marzo de 2024). *El mundo desperdicia más de 1.000 millones de platos de comida al día - informe de la ONU*. Obtenido de ONU: <https://www.unep.org/es/noticias-y-reportajes/comunicado-de-prensa/el-mundo-desperdicia-mas-de-1000-millones-de-platos-de>
- ONU. (2024). *Objetivo 12: Garantizar modalidades de consumo y producción sostenibles*. Obtenido de ONU: <https://www.un.org/sustainabledevelopment/es/sustainable-consumption-production/>
- OWASP . (2024). *OWASP SAMM*. Obtenido de OWASP: <https://owasp.org/www-project-samm/>
- OWASP. (2024). *Estándar de verificación de seguridad de aplicaciones (ASVS) de OWASP*. Obtenido de OWASP ORG: <https://owasp.org/www-project-application-security-verification-standard/>
- OWASP. (2024). *Quick start guide*. Obtenido de OWASP SAMM: <https://owaspsamm.org/guidance/quick-start-guide/>
- OWASP ASVS. (s.f.). *Estándar de verificación de seguridad de aplicaciones (ASVS) de OWASP*. Recuperado el Diciembre de 2024, de OWASP: <https://owasp.org/www-project-application-security-verification-standard/>

Pittet, S. (s.f.). *Los distintos tipos de pruebas de software*. Recuperado el 12 de 2024, de Atlassian Web Site: <https://www.atlassian.com/es/continuous-delivery/software-testing/types-of-software-testing>

Postman. (s.f.). *API platform*. Recuperado el 12 de 2024, de Postman: <https://www.postman.com/api-platform/>

Pytest. (s.f.). *Anatomy of a test*. Obtenido de Pytest: <https://docs.pytest.org/en/stable/explanation/anatomy.html#test-anatomy>

SAMM. (s.f.). *Requirements-driven Testing, Control Verification*. Recuperado el 2024, de OWASP SAMM: Verificación de control

Schleier, S., Holguera, C., Mueller, B., Willemsen, J., & Beckers, J. (2024). *Mobile Application Security Standard V2.1.0 (OWASP MASVS)*.

Snyk. (2024). *Ciclo de vida de desarrollo de software seguro (SSDLC)*. Obtenido de Snyk: <https://snyk.io/articles/secure-sdlc/>

Souppaya , M., Scarfone , K., & Dodson, D. (Febrero de 2022). *Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities*. Obtenido de NIST Computer Security Resource Center: <https://csrc.nist.gov/pubs/sp/800/218/final>

Taherdoost, H. (2022). *Understanding Cybersecurity Frameworks and Information Security Standards—A Review and Comprehensive Overview*. 07. doi:<https://doi.org/10.3390/electronics11142181>

World Economic Forum. (2024). *Global Cybersecurity Outlook 2024*.