REPÚBLICA DEL ECUADOR

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE POSGRADO

MAESTRÍA EN COMPUTACIÓN CON MENCIÓN EN SEGURIDAD INFORMÁTICA

TEMA:

"EVALUACIÓN DEL MARCO NIST SSDF PARA LA SEGURIDAD DEL CICLO DE DESARROLLO DE SOFTWARE EN CAJAS DE AHORRO DE REFIDER."

Trabajo de Investigación previo a la obtención del Título de Magister en Computación con mención en Seguridad de la Información.

AUTOR:

Ing. Estalin David Trujillo Casanova

DIRECTOR:

PHD. RAURA RUIZ JORGE GEOVANY.

Ibarra, 2026.

UNIVERSIDAD TÉCNICA DEL NORTE

DIRECCIÓN DE BIBLIOTECA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO	
APELLIDOS Y NOMBRES:	TRUJILLO CASANOVA ESTALIN DAVID

DATOS DE LA OBRA	
TÍTULO:	EVALUACIÓN DEL MARCO NIST SSDF PARA LA SEGURIDAD DEL CICLO DE DESARROLLO DE SOFTWARE EN CAJAS DE AHORRO DE REFIDER.
AUTOR (ES):	TRUJILLO CASANOVA ESTALIN DAVID
FECHA: DD/MM/AAAA	28/05/2026
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	( ) PREGRADO (x) POSGRADO
TITULO POR EL QUE OPTA:	Magíster en Computación con mención en Seguridad Infomática
DIRECTOR/ASESOR	PHD. RAURA RUIZ JORGE GEOVANY / PHD. GUEVARA VEGA CATHY PAMELA

2. CONSTANCIAS

El autor (es) manifiesta (n) que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es (son) el (los) titular (es) de los derechos patrimoniales, por lo que asume (n) la responsabilidad sobre el contenido de la misma y saldrá (n) en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 28 días del mes de mayo de 2026

(Firma)….

…….……..…

Nombre: Estalin David Trujillo Casanova

Ibarra, 12 de mayo de 2026

Dr. Jorge Gordón
Decano (e)
Facultad de Posgrado

ASUNTO: Conformidad con el documento final

Señor(a) Decano(a):

Nos permitimos informar a usted que revisado el Trabajo final de Grado "EVALUACIÓN DEL MARCO NIST SSDF PARA LA SEGURIDAD DEL CICLO DE DESARROLLO DE SOFTWARE EN CAJAS DE AHORRO DE REFIDER". del maestrante TRUJILLO CASANOVA ESTALIN DAVID, de la Maestría de Computación Mención Seguridad Informática en Línea (CSI-L), certificamos que han sido acogidas y satisfechas todas las observaciones realizadas.

Atentamente,

	Apellidos y Nombres	Firma
Director/a	Phd. RAURA RUIZ JORGE GEOVANY	Validar únicamente en FirmaEC. Firmado electrónicamente por: JORGE GEOVANNY RAURA RUIZ
Asesor/a	Phd. CATHY PAMELA GUEVARA VERA	Validar únicamente en FirmaEC. Firmado electrónicamente por: CATHY PAMELA GUEVARA VEGA

DEDICATORIA

Dedico el presente trabajo de investigación, en primer lugar, a mi esposa, por su amor incondicional, paciencia y apoyo constante durante todo este proceso académico. Su comprensión y motivación fueron fundamentales para continuar adelante en los momentos más difíciles.

A mis hijos, quienes representan mi mayor inspiración y la razón principal para seguir superándome cada día. Este logro también es para ustedes, como ejemplo de esfuerzo, dedicación y perseverancia.

A mi madre, por sus enseñanzas, sacrificios y valores inculcados a lo largo de mi vida. Gracias por ser el pilar fundamental de mi formación humana y profesional.

AGRADECIMIENTO

Expreso mi más sincero agradecimiento a Dios, por permitirme culminar esta importante etapa académica, brindándome fortaleza, sabiduría y perseverancia para superar cada desafío presentado durante el desarrollo de esta investigación.

De manera especial, agradezco a mi director de tesis, por su guía, acompañamiento y recomendaciones durante el desarrollo de este trabajo investigativo, permitiendo fortalecer cada una de las etapas de la investigación.

UNIVERSIDAD TÉCNICA DEL NORTE
FACULTAD DE POSGRADO

MAESTRÍA EN COMPUTACIÓN CON MENCIÓN EN SEGURIDAD INFORMÁTICA

"EVALUACIÓN DEL MARCO NIST SSDF PARA LA SEGURIDAD DEL CICLO DE DESARROLLO DE SOFTWARE EN CAJAS DE AHORRO DE REFIDER."

Autor: Estalin David Trujillo Casanova
Tutor: Phd. Geovany Raura Ruiz
Año: 2025

RESUMEN

Este estudio aborda la seguridad en el desarrollo de software dentro de las instituciones financieras socias de REFIDER. La fase diagnóstica reveló un nivel de madurez incipiente (0.95/5) caracterizado por procesos empíricos y la ausencia de políticas formales. Ante los riesgos críticos detectados, como la inyección de código y fallos criptográficos en sistemas legados, el objetivo de la investigación fue establecer un proceso de desarrollo seguro mediante la implementación y evaluación del marco NIST SSDF. El enfoque metodológico adoptado fue de carácter cualitativo y cuantitativo. Se adaptaron las prácticas del marco NIST a las necesidades locales y se desarrolló un prototipo funcional de un módulo financiero aplicando principios de "seguridad por diseño". Los controles técnicos implementados incluyeron la gestión de identidades mediante tokens JWT, el cifrado de credenciales con BCrypt, transacciones ACID para asegurar la integridad de los datos y una validación estricta de entradas para mitigar ataques OWASP. Los resultados confirmaron una mejora significativa en cuanto a la seguridad en el desarrollo de software de la institución, logrando una reducción del 80% en las vulnerabilidades críticas y elevando el cumplimiento de

las prácticas del marco del 17.75% al 71.5%. Asimismo, se optimizó la capacidad de respuesta ante incidentes, reduciendo el tiempo medio de reparación (MTTR) de 96 a 24 horas. En conclusión, la adopción del NIST SSDF no solo minimiza los riesgos operativos y protege los activos financieros de los socios, sino que profesionaliza el ciclo de vida del software, asegurando la sostenibilidad tecnológica y el cumplimiento normativo exigido por los organismos de control.

Palabras claves: Ciberseguridad, NIST SSDF, SDLC seguro, Sector Financiero, OWASP, Gestión de Vulnerabilidades.

ABSTRACT

This study addresses software development security within the financial institutions partnered with REFIDER. The diagnostic phase revealed an incipient maturity level (0.95/5), characterized by empirical processes and a lack of formal policies. In response to critical risks identified—such as code injection and cryptographic failures in legacy systems—the research aimed to establish a secure development process through the implementation and evaluation of the NIST SSDF framework.

The methodological approach adopted was both qualitative and quantitative. NIST framework practices were adapted to local requirements, and a functional prototype of a financial module was developed applying "security by design" principles. Implemented technical controls included identity management via JWT tokens, credential encryption using BCrypt, ACID transactions to ensure data integrity, and strict input validation to mitigate OWASP attacks.

The results confirmed a significant improvement in the institution's software development security, achieving an 80% reduction in critical vulnerabilities and increasing framework practice compliance from 17.75% to 71.5%. Furthermore, incident response capabilities were optimized, reducing the Mean Time to Repair (MTTR) from 96 to 24 hours. In conclusion, the adoption of NIST SSDF not only minimizes operational risks and protects members' financial assets but also professionalizes the software life cycle, ensuring technological sustainability and regulatory compliance demanded by oversight bodies.

Keywords:

Cybersecurity, NIST SSDF, Secure SDLC, Financial Sector, OWASP, Vulnerability Management.

Contenido

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE 2
DEDICATORIA 5
AGRADECIMIENTO 6
RESUMEN 7
ABSTRACT 9
1. EL PROBLEMA 13
1.1. Problema de investigación 13
1.2. Interrogantes de la investigación 15
1.3. Objetivos de la investigación 15
1.3.1. Objetivo general 15
1.3.2. Objetivos específicos 16
1.4. Hipótesis de trabajo 16
1.5. Hipótesis alternativa 16
1.6. Categorización de variables 16
1.7. Justificación 17
2. MARCO REFERENCIAL 19
2.1. Antecedentes 19
2.2. Marco Teórico 29
2.2.1. SDLC 29
2.2.2. Tipos de modelo SDLC 31
2.2.3. La importancia de la Seguridad en el Ciclo de Desarrollo de Software (SDLC) 36
2.2.4. Amenazas y Vulnerabilidades en el SDLC 37
2.2.5. Consecuencias de las Fallas de Seguridad en el Software 37
2.2.6. El Marco NIST de Seguridad del Desarrollo de Software (SSDF) 38
2.2.6.1. Descripción General del SSDF 38
2.2.6.2. Relación del SSDF con Otros Marcos y Estándares de Seguridad (OWASP, ISO/IEC 27034) 41
2.2.7. Marco Legal y Normativo Aplicable 42
3. MARCO METODOLOGICO 44
3.1. Descripción del área de estudio / Descripción del grupo de estudio 44
3.2. Enfoque y tipo de investigación 45
3.3. Procedimientos de investigación 46

3.4. Consideraciones bioéticas 51
4. RESULTADOS 52
4.1. Evaluación del estado actual del SDLC en las cajas de ahorro de REFIDER 52
4.2. Nivel de alineación con el marco NIST SSDF 53
4.3. Controles de seguridad implementados en el módulo financiero 58
4.3.1. Descripción de los controles implementados 59
4.4. Implementación del módulo financiero seguro basado en SSDF (prototipo) 61
4.4.1 Preparar la Organización (PO) 61
4.4.2 Proteger el Software (PS) 62
4.4.3 Producir Software Bien Asegurado (PW) 64
4.4.4 Responder a Vulnerabilidades (RV) 67
4.4.5 Comparación del nivel de madurez antes y después de aplicar SSDF 69
4.5. Análisis comparativo del estado AS-IS y TO-BE basado en NIST SSDF 72
4.6. Evaluación del SSDF 74
4.7. Informe sobre la aplicación del Juicio de Expertos en el caso de estudio. 75
4.7.1 Tabulación de resultados 76
4.8. Análisis de resultados de acuerdo al Juicio de Expertos y conclusión respecto a la hipótesis de trabajo 80
5. CONCLUSIONES Y RECOMENDACIONES 82
5.1. Conclusiones 82
5.2. Recomendaciones 83
REFERENCIAS 87

INDICE DE FIGURAS

Figura 1 Verificación IEEE Explorer 20
Figura 2 Ciclo de Desarrollo Seguro (servicenow, s.f.) 29
Figura 3 Modelo cascada (bdtask, s.f.) 32
Figura 4 Modelo en V (geeksforgeeks, s.f.) 32
Figura 5 Modelo espiral (teachingagile, s.f.) 33
Figura 6 Flujo de trabajo en Scrum (productiviza, s.f.) 34
Figura 7 Metodología XP (sinnaps, s.f.) 35
Figura 8 Modelo DevOps (aws.amazon, s.f.) 36
Figura 9 Integración del SSDF en ciclo de desarrollo (SDLC) Fuente Propia 54

Figura 10 Configuración de seguridad mediante Spring Security Fuente Propia 63
Figura 11 Implementación de autenticación JWT 64
Figura 12 Gestión de Transacciones seguras Fuente Propia 65
Figura 13 Validación de entradas mediante DTO Fuente Propia 66
Figura 14 Servicio de auditoría y registro de eventos Fuente Propia 68

INDICE DE TABLAS

Tabla 1 NIST SSDF (Autoría Propia) 29
Tabla 2 Comparación de modelos SDLC Fuente Propia 36
Tabla 3 Población de estudio Fuente Propia 45
Tabla 4 Valoraciones 48
Tabla 5 Vulnerabilidades encontradas Fuente Propia 53
Tabla 6 Matriz de evaluación inicial fuente propia 57
Tabla 7 Promedio de evaluación Fuente Propia 57
Tabla 8 Nivel de cumplimiento Fuente Propia 57
Tabla 9 Controles de seguridad implementados Fuente Propia 59
Tabla 10 Control Implementado PO Fuente Propia 62
Tabla 11 Controles Implementados PS Fuente Propia 62
Tabla 12 Implementación de autenticación JWT. 63
Tabla 13 Controles implementados PW Fuente Propia 64
Tabla 14 Controles implementados RV Fuente Propia 67
Tabla 15 Implantaciones Técnicas Fuente Propia 69
Tabla 16 Comparación Inicial y Final Fuente Propia 71
Tabla 17 Valores comparativos Inicio y Fin Fuente Propia 71
Tabla 18 (AS-IS) y (TO-BE) Fuente Propia 73
Tabla 19 Criterios de Evaluación para Juicio de expertos Fuente Propia 74
Tabla 20 Calificación para Juicio de expertos Fuente Propia 75
Tabla 21 Expertos Fuente Propia 75
Tabla 22 Matriz de cumplimiento Fuente Propia 76
Tabla 23 Resultados Juicio de expertos Fuente Propia 76

CAPITULO I

1. EL PROBLEMA

1.1. Problema de investigación

La creciente sofisticación de las ciberamenazas y su impacto potencial en instituciones financieras como las cajas de ahorro han resaltado la criticidad de integrar la seguridad de manera robusta en el ciclo de desarrollo de software (SDLC). Tradicionalmente, la seguridad a menudo se consideraba una fase aislada, posterior a la codificación, lo que generaba la identificación tardía de vulnerabilidades costosas de remediar y susceptibles a la explotación maliciosa (McGraw, 2006). Este enfoque reactivo resulta particularmente problemático en el sector financiero, donde la confidencialidad, integridad y disponibilidad de la información de los clientes son primordiales y están sujetas a estrictas regulaciones.

Hitos clave en los avances de seguridad Los puntos de inflexión importantes han definido el desarrollo de las técnicas de seguridad de software:

1. Introducción del ciclo de vida de desarrollo de software seguro (SDL): Microsoft propuso por primera vez el SDL, que combina la seguridad en cada etapa del desarrollo de software, a principios de la década de 2000. Muchas empresas han adoptado este enfoque para gestionar metódicamente la seguridad desde el diseño hasta la implementación (Microsoft, 2004).

2. Publicación de OWASP Top Ten: publicada por primera vez en 2003, la publicación OWASP Top Ten se ha convertido en una herramienta vital para detectar y corregir las debilidades típicas de las aplicaciones web. Ha ayudado a estandarizar las prácticas de seguridad en toda la industria (OWASP, 2021).

3. Desarrollo de algoritmos criptográficos avanzados: La introducción del Estándar de cifrado avanzado (AES) en 2001 proporcionó un estándar criptográfico más seguro y efectivo, que ha sido ampliamente adoptado para el desarrollo de la seguridad de los datos (Thomas et al., 2018)

4. Auge de DevSecOps: DevSecOps, o la integración de la seguridad en los métodos DevOps, surgió por primera vez en la década de 2010. Este método enfatiza la integración y automatización constantes de la seguridad en todo el ciclo de vida del desarrollo de software, mejorando así la agilidad y la seguridad de la entrega de software (Kim et al., 2015; Sánchez-Gordón y Colomo-Palacios, 2020).

La inclusión financiera digital se constituye en una alternativa. El uso de canales digitales coadyuva a una mayor inclusión financiera, entendida como el acceso a productos financieros útiles y asequibles que satisfagan las necesidades -transacciones, pagos, ahorros, crédito y seguro- de personas y empresas, pero además considerando un enfoque de inclusión social, la inclusión financiera promueve la igualdad de oportunidades entre los miembros de una sociedad, sobre todo, de quienes se encuentran en una situación de vulnerabilidad (Rivera & Guerra, 2019).

Las cajas de ahorro y crédito en Ecuador, como las socias de REFIDER, desempeñan un papel fundamental en la inclusión financiera y el desarrollo económico local. Sin embargo, su manejo de datos financieros sensibles y su creciente adopción de tecnologías digitales las convierten en objetivos atractivos para los ciberdelincuentes. Un software desarrollado sin una adecuada consideración de la seguridad en cada etapa de su ciclo de vida puede introducir vulnerabilidades que expongan a estas instituciones y a sus miembros a riesgos significativos, incluyendo pérdidas financieras, daño a la reputación e incumplimiento normativo (Brown et al., 2022).

Las cajas de ahorro socias de REFIDER desempeñan un papel crucial en la inclusión financiera de comunidades locales, pero muchas de ellas operan con sistemas transaccionales obsoletos o carecen de ellos por completo. Esta situación genera graves riesgos de seguridad de la información, ya que los sistemas legados son vulnerables a ataques cibernéticos, como la inyección de código y fallos criptográficos, que pueden comprometer los datos financieros y personales de los socios. La falta de sistemas transaccionales modernos también limita la capacidad de las cajas para ofrecer servicios financieros eficientes y seguros, lo que afecta su competitividad y la confianza de sus socios. Este estudio propone el desarrollo de un módulo como prototipo, siguiendo las mejores prácticas de desarrollo de software seguro.

1.2. Interrogantes de la investigación

- ¿Cómo se pueden adaptar las prácticas del marco NIST SSDF para satisfacer las necesidades específicas del desarrollo de software en las cajas de ahorro socias de REFIDER, considerando sus limitaciones y características particulares?
- ¿Cómo se puede implementar un módulo financiero seguro para las cajas de ahorro socias de REFIDER, aplicando los principios del marco NIST SSDF en cada etapa del ciclo de vida del desarrollo de software?
- ¿Cómo se puede validar la seguridad del módulo desarrollado en cada fase del ciclo de vida, utilizando métricas de seguridad informática?

1.3. Objetivos de la investigación

1.3.1. Objetivo general

Establecer un proceso de desarrollo de software seguro para las cajas de ahorro de REFIDER, mediante la implementación y evaluación del marco NIST SSDF, para minimizar vulnerabilidades y el impacto de su explotación.

1.3.2. Objetivos específicos

- Adaptar las prácticas del marco NIST SSDF a las necesidades del desarrollo de software en las cajas de ahorro de REFIDER.
- Implementar un módulo financiero para las cajas de ahorro de REFIDER, aplicando los principios del marco NIST SSDF (Secure Software Development Framework) en cada etapa del ciclo de desarrollo.
- Validar la seguridad del módulo desarrollado en cada fase del ciclo de vida, utilizando métricas de seguridad informática.

1.4. Hipótesis de trabajo

Establecer un proceso de desarrollo de software seguro para las cajas de ahorro de REFIDER, mediante la implementación y evaluación del marco NIST SSDF permitirá minimizar vulnerabilidades y el impacto de su explotación.

1.5. Hipótesis alternativa

Establecer un proceso de desarrollo de software seguro para las cajas de ahorro de REFIDER, mediante la implementación y evaluación del marco NIST SSDF no permitirá minimizar vulnerabilidades y el impacto de su explotación.

1.6. Categorización de variables

Variable independiente: Proceso de desarrollo de software seguro para las cajas de ahorro de REFIDER mediante la implementación y evaluación del marco NIST SSDF.

Variable dependiente: Vulnerabilidades y el impacto de su explotación.

1.7. Justificación

La seguridad de la información constituye un pilar esencial dentro del ámbito de las tecnologías de la información, aplicable a instituciones de cualquier naturaleza y tamaño. Este concepto se fundamenta en la necesidad de proteger los datos frente a accesos no autorizados, así como en garantizar su integridad a lo largo de todo su ciclo de vida.

Dentro de este campo, se integran diversas prácticas y técnicas, tales como la encriptación de datos, la informática forense y la gestión eficiente de claves criptográficas, las cuales contribuyen de manera significativa a la salvaguarda de la información en múltiples plataformas y aplicaciones organizacionales.

En la actualidad, las organizaciones a nivel global realizan inversiones sustanciales en tecnologías orientadas a la ciberdefensa, con el propósito de proteger activos estratégicos, tales como su reputación institucional, el capital intelectual y los datos sensibles de sus clientes.

La creciente importancia de la seguridad informática en el sector financiero, especialmente en instituciones que manejan datos sensibles de sus socios, como las cajas de ahorro de REFIDER. Estas instituciones, pilares de la economía popular y solidaria, enfrentan desafíos únicos en la protección de su información debido a la obsolescencia de sus sistemas y la falta de recursos para implementar soluciones de seguridad robustas.

La ausencia de un ciclo de desarrollo de software seguro (SDLC) en estas cajas de ahorro las expone a riesgos significativos, incluyendo:

- Vulnerabilidades: Sistemas obsoletos son susceptibles a ataques que pueden comprometer la confidencialidad, integridad y disponibilidad de los datos financieros.

- Pérdida de confianza: Incidentes de seguridad pueden erosionar la confianza de los socios, afectando la estabilidad y sostenibilidad de las cajas de ahorro.
- Impacto económico y social: La materialización de riesgos cibernéticos puede tener consecuencias devastadoras para las comunidades a las que sirven estas instituciones.

En este contexto, la evaluación y adaptación del marco NIST SSDF se presenta como una solución viable para fortalecer el SDLC en las cajas de ahorro de REFIDER. Este marco, reconocido internacionalmente, proporciona un conjunto de prácticas y directrices para desarrollar software seguro, minimizando vulnerabilidades y mitigando riesgos.

Esta tesis busca cerrar una brecha crítica en la seguridad informática de las cajas de ahorro de REFIDER, promoviendo la adopción de prácticas de desarrollo de software seguro que protejan sus datos, fortalezcan su confianza y aseguren su sostenibilidad a largo plazo.

CAPITULO II

2. MARCO REFERENCIAL

2.1. Antecedentes

Esta sección presenta una revisión exhaustiva de la literatura científica existente directamente relacionada con la evaluación de marcos de seguridad en el ciclo de desarrollo de software (SDLC) y su aplicación en el sector financiero, particularmente en instituciones similares a las cajas de ahorro. El análisis de estos antecedentes permitirá identificar las brechas en la investigación actual y justificar la relevancia de este estudio en el contexto específico de la evaluación del NIST SSDF en las Cajas de Ahorro de REFIDER.

En primera instancia, se determinó una cadena de búsqueda que permita contestar a la siguiente interrogante de investigación: ¿Cómo se puede implementar un módulo financiero seguro para las cajas de ahorro socias de REFIDER, aplicando los principios del marco NIST SSDF en cada etapa del ciclo de vida del desarrollo de software?

Para encontrar los artículos que respondan a esta interrogante, se definió la siguiente cadena de búsqueda: ((("All Metadata":Secure Software Development Practices) OR ("All Metadata":NIST SSDF) OR ("All Metadata":SP 800-218)) AND (("All Metadata":financial services) OR ("All Metadata":banking)))

La cadena de búsqueda fue corrida en la base de datos de IEEE Xplore (IEEEX), misma que mostró un total de 40 publicaciones. Se utilizó como criterio de inclusión los trabajos relacionados con el manejo de sistemas de seguridad de la información publicados desde el año 2018 hasta el año 2025, que estén en idioma inglés y que hagan referencia al desarrollo seguro como se indica en la Figura 1.

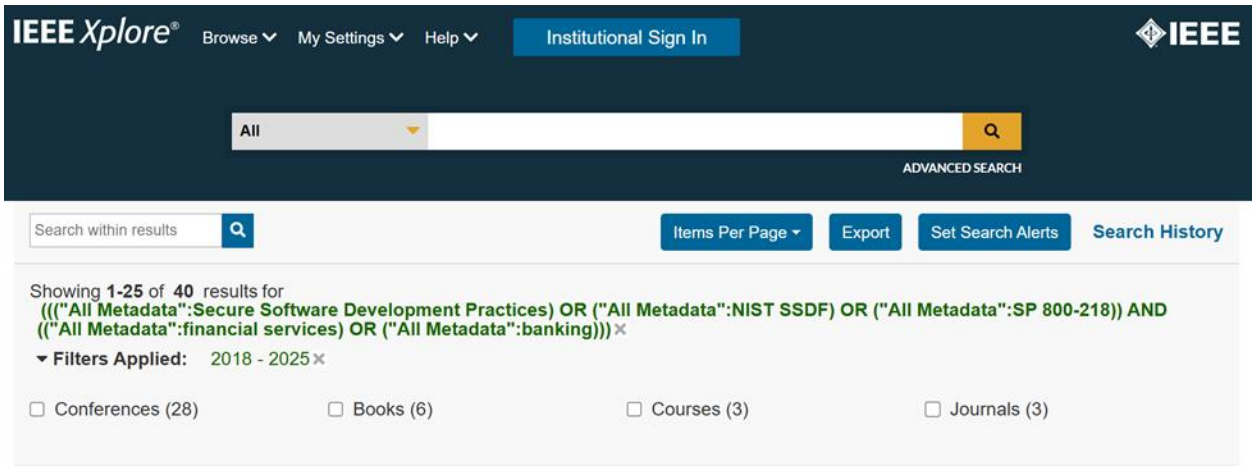


Figura 1 Verificación IEEE Explorer

Posteriormente se analizó un artículo científico, el cual realizaba un estudio de literatura sobre el tema, en el cual se pudo encontrar que se realizó la investigación de 10 artículos, debido a que cumplen los criterios de inclusión y exclusión antes mencionados. El análisis de los artículos se detalla a continuación.

Según (Khari, Vaishali, & Kumar, 2016) los usuarios que esperan mayor confiabilidad, seguridad y facilidad de uso exigen un software de mayor calidad. El avance de las nuevas tecnologías conlleva mayores expectativas. Los usuarios desean acceder a un software libre de errores y fácil de usar. Para lograrlo, se debe considerar un marco estructurado o un ciclo de vida bien definido. El ciclo de vida no solo debe indicar las funciones que debe realizar, sino que también debe proporcionar un alto nivel de seguridad. El objetivo de este documento es integrar el ciclo de vida de desarrollo de software (SDLC) tradicional con las propiedades de riesgo y seguridad en un solo ciclo de vida. Al igual que en el SDLC tradicional, las pruebas de seguridad se realizan al final, pero se deben considerar los factores de seguridad y riesgo en cada etapa para prevenir errores importantes en etapas posteriores.

Según (Talukder.,2009), una aplicación se protege mediante seguridad perimetral in vitro. Por ello, la seguridad se considera un requisito no funcional en el ciclo de vida del desarrollo de software (SDLC). En la Internet de Próxima Generación (NGI), donde todas las aplicaciones estarán conectadas en red, la seguridad debe ser in vivo; la seguridad debe ser funcional dentro de la aplicación. Las aplicaciones que se ejecutan en cualquier dispositivo, ya sea móvil o fijo, deben ser conscientes de la seguridad mediante el Ciclo de Vida de Desarrollo de Software. Presentan una herramienta llamada Suraksha, que incluye Security Designers' Workbench y Security Testers' Workbench, y que ayuda a los desarrolladores a crear aplicaciones con conciencia de seguridad.

Para (Khan, Khan, Khan, & llyas, 2022) la seguridad es uno de los aspectos más críticos de la calidad del software. Se refiere al proceso de creación y desarrollo de software que garantiza la integridad, confidencialidad y disponibilidad de su código, datos y servicios. Las organizaciones de desarrollo de software tratan la seguridad como un asunto secundario y, como resultado, continúan enfrentándose a amenazas. Incorporar la seguridad en cualquier nivel del ciclo de vida del desarrollo de software (SDLC) se ha convertido en un requisito urgente. Se han propuesto y desarrollado diversas metodologías, estrategias y modelos para abordar la seguridad del software, pero solo algunos ofrecen evidencia fiable para la creación de aplicaciones seguras. Por otro lado, los problemas de seguridad del software no se han abordado adecuadamente, y la integración de procedimientos de seguridad en el ciclo de vida del desarrollo de software (SDLC) sigue siendo un desafío. Para comprender los riesgos y las prácticas de seguridad del software para diseñar mejores métodos de desarrollo de software seguro, los autores realizaron una revisión sistemática de la literatura (SLR). Con base en los criterios de inclusión, exclusión y evaluación de calidad, se seleccionaron 121 estudios. Este estudio identificó 145 riesgos de seguridad y 424 mejores prácticas que ayudan a las organizaciones de desarrollo de software a gestionar la seguridad en

cada fase del ciclo de vida del desarrollo de software (SDLC). Para lograr un SDLC seguro, el estudio prescribió diferentes actividades de seguridad que deben seguirse en cada fase. La integración exitosa de estas actividades minimiza el esfuerzo, el tiempo y el presupuesto, a la vez que ofrece aplicaciones de software seguras. Los hallazgos de este estudio ayudan a las organizaciones de desarrollo de software a mejorar el nivel de seguridad de sus productos y a optimizar su eficiencia. Esto también aumentará la concienciación de los desarrolladores sobre las prácticas de desarrollo seguro.

En el estudio de (Maheshwari & Prasanna, 2017) se realiza la evaluación de riesgos y el modelado de amenazas se realizan con diferentes propósitos. De acuerdo a los autores, la integración de ambos procesos limita el riesgo de los sistemas de software. Incorporar la seguridad en todas las fases del ciclo de vida del desarrollo de software es una tarea tediosa en muchas organizaciones. Como principales hallazgos del estudio, se menciona que la fase de diseño del ciclo de vida del desarrollo de software (SDLC), se identifica y detecta el 50 % de los defectos de software. La mayoría de los ataques de seguridad ocurren en la capa de aplicación. Este documento explica el uso combinado de la evaluación de riesgos y el modelo de amenazas para comprender el riesgo de seguridad de una aplicación. También analizamos cómo el modelo puede identificar amenazas y cómo priorizarlas por categoría. Finalmente, recomendamos comprender el riesgo de detección y crear un entorno justo para reducir la probabilidad de que los atacantes cometan actos delictivos.

Según (Alioune Diallo, Aicha War, Moustapha Awwalou Diouf, Jordan Samhi, Steven Arzt, Tegawendé F. Bissyandé y Jacques Klein., 2025) , el auge de la banca móvil en los estados de la Unión Económica y Monetaria del África Occidental (UEMOA/WAEMU). Ante la proliferación de aplicaciones de banca móvil (MBAs) que permiten transferencias, pagos y consultas, los autores realizan una evaluación de seguridad mediante análisis estático de 59 aplicaciones pertenecientes

a instituciones financieras listadas por el Banco Central de los Estados de África Occidental (BCEAO). El objetivo es identificar vulnerabilidades de código y rastrear su evolución a través de actualizaciones para comparar el nivel de seguridad con aplicaciones de países desarrollados.

En el estudio de (N. Srivastava, N. Singh y K. Suhail., 2023), se investiga los riesgos de seguridad del software en la industria bancaria, centrando su atención exclusivamente en la fase de diseño del Ciclo de Vida del Desarrollo de Software (SDLC). Los autores argumentan que la mayoría de las brechas de seguridad bancaria no se deben a fallos aleatorios, sino a defectos estructurales y arquitectónicos introducidos antes de escribir la primera línea de código. El estudio identifica los principales vectores de riesgo en esta etapa y propone un marco conceptual para mitigar las amenazas antes de que el software pase a la fase de desarrollo, reduciendo drásticamente los costos de reparación.

El estudio de (Alejandro Calderón, Muqaibil.M,Jalil.S, 2020), presenta una evaluación empírica masiva y sistemática de la seguridad de las aplicaciones de banca móvil para Android a escala global. Los autores recopilan y analizan miles de aplicaciones bancarias de diferentes países y continentes. Utilizando herramientas avanzadas de análisis estático y dinámico, el artículo examina la presencia de vulnerabilidades críticas, malas prácticas de codificación, fugas de datos y violaciones de privacidad. El objetivo es proporcionar un panorama realista y cuantitativo sobre si los bancos globales están protegiendo adecuadamente los activos financieros y la información personal de sus usuarios en el ecosistema móvil.

Los autores (Nilay Yıldırım y Asaf Varol., 2019), examinan de forma exhaustiva las amenazas de seguridad y las medidas preventivas implementadas en los sistemas de banca móvil y en línea. Destacan que, a medida que las entidades financieras añaden nuevas funcionalidades para mejorar la experiencia de usuario, la superficie de ataque se expande significativamente. La investigación

recopila y analiza vectores de ataque comunes, como de ingeniería social y evalúa el comportamiento de diversas aplicaciones financieras en sistemas operativos móviles. El propósito central es mapear las brechas de seguridad emergentes asociadas a estas innovaciones para proponer controles de mitigación eficientes.

Para (Hussain Al-Matouq, Sajjad Mahmood, Mohammad Alshayeb y Mahmood Niazi, 2020), la problemática de que los fallos de diseño de software siguen siendo una de las principales fuentes de vulnerabilidades explotables en producción. Para solucionar esto, los autores desarrollan un Modelo de Madurez para el Diseño de Software Seguro (SSDM-MM). La investigación utiliza un enfoque de estudio multivocal (Multivocal Literature Review - MLR), el cual combina la literatura académica formal con la literatura gris (estándares industriales, libros blancos, blogs de expertos y reportes de la industria). El modelo resultante identifica, clasifica y valida un conjunto de prácticas críticas que las organizaciones deben adoptar de forma progresiva para garantizar la seguridad desde la arquitectura de sus sistemas.

Según (S. Moyo y E. Mnkandla., 2020), la problemática de que las metodologías de desarrollo de software seguro tradicionales (como Microsoft SDL, CLASP o Touchpoints) están diseñadas para grandes organizaciones con equipos multidisciplinarios y comités de seguridad dedicados. Para resolver esto, los autores proponen una nueva metodología ligera de desarrollo de software para un solo programador (Solo-SDM) que integra prácticas óptimas de seguridad sin sobrecargar el flujo de trabajo con burocracia o documentación excesiva. El modelo busca democratizar el desarrollo seguro, garantizando que un programador independiente pueda entregar software resiliente a un costo operativo viable.

Conclusión de la revisión de literatura:

En base al análisis de los estudios citados, se evidencia que la seguridad en el desarrollo de software ha evolucionado desde enfoques reactivos hacia modelos preventivos integrados durante todo el ciclo de vida del desarrollo de software (SDLC). Los principales hallazgos identificados permiten concluir que la incorporación temprana de prácticas de seguridad constituye un factor crítico para reducir vulnerabilidades y fortalecer la resiliencia de aplicaciones financieras.

Los estudios analizados coinciden en que uno de los principales factores de éxito radica en integrar la seguridad desde las primeras fases del diseño arquitectónico, aplicando enfoques como "Shift Left Security", modelado de amenazas y validación continua. Autores como V Maheshwari y N. Srivastava destacan que la evaluación de riesgos y el uso de metodologías como STRIDE y PASTA permiten identificar amenazas tempranamente, disminuyendo defectos de seguridad antes de llegar a producción.

Asimismo, la revisión bibliográfica demuestra que el marco National Institute of Standards and Technology SSDF proporciona una estructura adecuada para fortalecer procesos de desarrollo seguro, especialmente en sistemas financieros que manejan información sensible y transacciones críticas. Los trabajos de Rafiq Ahmad Khan resaltan la importancia de incorporar prácticas de protección continua, monitoreo y gestión de vulnerabilidades dentro del SDLC para minimizar riesgos operativos y cibernéticos.

Otro hallazgo relevante identificado en la literatura es que la mayoría de vulnerabilidades críticas no se originan durante el mantenimiento del software, sino desde las etapas iniciales de desarrollo. Esto es evidenciado por Alioune Diallo, quienes concluyen que muchas fallas de

seguridad son introducidas durante la construcción inicial del software debido a la ausencia de prácticas formales de desarrollo seguro.

De igual forma, varios estudios evidencian que las aplicaciones financieras representan uno de los entornos con mayor exposición a amenazas de ciberseguridad. Investigaciones como las de Alejandro Calderón y Nilay Yıldırım identifican vulnerabilidades críticas relacionadas con validaciones insuficientes, debilidades en autenticación, ausencia de cifrado robusto, exposición del código fuente y ataques de interceptación de comunicaciones.

ARTICULO NO.	AUTOR/ES	AÑO DE PUBLICACIÓN	ENTORNO TECNOLOGICO	FACTORES DE ÉXITO	FACTORES DE RIESGO/HALLAZGOS CLAVES
1	Manju Khari, Vaishali y Prabhat Kumar	2016	Seguridad	Mayor confiabilidad, seguridad y facilidad de uso exigen un software de mayor calidad	El avance de las nuevas tecnologías conlleva mayores expectativas
2	Asoke K. Talukder, Vineet Kumar Maurya, Babu G. Santhosh, Ebenezer Jangam, Sekhar V. Muni	2009	Desarrollo	Proteger mediante seguridad perimetral in vitro. Por ello, la seguridad se considera un requisito no funcional en el ciclo de vida del desarrollo de software	También presentamos una herramienta llamada Suraksha, que incluye Security Designers' Workbench y Security Testers' Workbench, y que ayuda a los desarrolladores
3	V Maheshwari, M Prasanna	2016	Seguridad	La evaluación de riesgos y el modelado de amenazas. Incorporar la seguridad en todas las fases del ciclo de	Se identifica y detecta el 50 % de los defectos de software

				vida del desarrollo de software.	
4	Rafiq Ahmad Khan, Siffat Ullah Khan, Habib Ullah Khan	2022	Seguridad	Desarrollo de Software seguro según la NIST	
5	Alioune Diallo, Aicha War, Moustapha Awwalou Diouf, Jordan Samhi, Steven Arzt, Tegawendé F. Bissyandé y Jacques Klein	2025	Desarrollo	La infraestructura móvil ha permitido la inclusión financiera donde la banca física es escasa.	La mayoría de los fallos de seguridad se introducen durante la creación inicial del artefacto y no durante su evolución, lo que indica falta de prácticas de Security.
6	N. Srivastava, N. Singh y K. Suhail.	2023	Seguridad	Implementar metodologías como STRIDE o PASTA formalmente durante la fase de diseño arquitectónico.	El diseño de mecanismos de control de acceso defectuosos (como la falta de validación estricta de tokens en la arquitectura) permite ataques de escalada de privilegios.
7	Alejandro Calderón, Muqaibil.M,Jalil. S,	2020	Seguridad	El éxito en la protección de las comunicaciones radica en el uso correcto del anclaje de certificados, impidiendo ataques donde un tercero intercepta la conexión.	Un porcentaje alarmantemente alto de las aplicaciones bancarias evaluadas a nivel mundial contenía al menos una vulnerabilidad de seguridad de nivel "alto" o "crítico".

8	Nilay Yıldırım y Asaf Varol.	2019	Seguridad	Despliegue mandatorio de conexiones seguras basadas en SSL/TLS para anular la efectividad de herramientas de interceptación de red.	El 60% de las aplicaciones bancarias en Android evaluadas no ocultaban ni protegían su código fuente, lo que las expone directamente a técnicas de ingeniería inversa, clonación de apps y análisis de código por parte de atacantes.
9	Hussain Al-Matouq, Sajjad Mahmood, Mohammad Alshayeb y Mahmood Niazi,	2020	Desarrollo Seguro	Establecer qué debe proteger el software antes de estructurar los diagramas de componentes.	El estudio multivocal reveló que muchas prácticas de diseño seguro defendidas en la literatura académica son ignoradas por la industria debido a su complejidad teórica o falta de escalabilidad comercial.
10	S. Moyo y E. Mnkandla.	2020	Seguridad	Reemplazar las revisiones de diseño complejas por listas de verificación personalizadas basadas en el tipo de aplicación (por ejemplo, asegurar el manejo de JWT o transacciones	El hallazgo de que exigirle a un solo desarrollador que cumpla con todos los estándares tradicionales de seguridad provoca el abandono total de las prácticas seguras debido a la sobrecarga de trabajo y la presión por los

				ACID desde el inicio si es software financiero).	tiempos de entrega.

Tabla 1 NIST SSDF (Autoría Propia)

2.2.Marco Teórico

2.2.1. SDLC

El ciclo de vida del desarrollo de software (SDLC) es un proceso rentable y eficiente en términos de tiempo empleado por los equipos de desarrollo para diseñar y crear software de alta calidad. El objetivo del SDLC es minimizar los riesgos del proyecto por medio de una planificación anticipada que permita que el software cumpla las expectativas del cliente durante la fase de producción y posteriormente. Esta metodología establece una serie de pasos que dividen el proceso de desarrollo de software en tareas que se pueden asignar, completar y medir. (Amazon, s.f.)

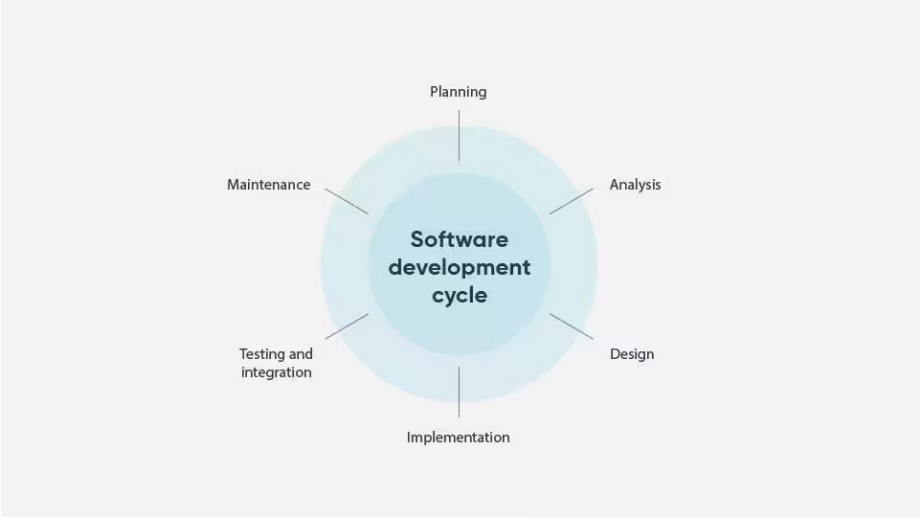


Figura 2 Ciclo de Desarrollo Seguro (servicenow, s.f.)

Según (redhat, s.f.) El ciclo de vida de desarrollo del software (SDLC) es un marco que se utiliza para desarrollar, implementar y mantener el software. El marco implica la ejecución de varias tareas y actividades que se dividen en seis u ocho etapas para mejorar la calidad del software haciendo énfasis en el proceso. El propósito de esto es que sea posible medir y analizar el sistema, incorporar mejoras y, a su vez, supervisar el progreso y los costos.

Estas son las etapas del SDLC según (redhat, s.f.):

- Planificación: determinar el alcance y la finalidad del software
- Análisis de los requisitos: define las funciones que debe ejecutar el software
- Diseño: decidir los parámetros clave, como la arquitectura, las plataformas y las interfaces de usuario
- Desarrollo: crear e implementar el software
- Documentación: producir la información para que los usuarios y las partes interesadas puedan utilizar el sistema
- Pruebas: verificar que el software cumpla con los requisitos
- Implementación: poner el software a disposición de los usuarios
- Mantenimiento: solucionar los errores y los puntos vulnerables que se descubran en el sistema

A simple vista, el SDLC y la gestión del ciclo de vida de las aplicaciones (ALM) parecen muy similares: ambos se encargan de desarrollar y gestionar el sistema. Sin embargo, el ciclo de vida de desarrollo del software puede considerarse un subconjunto de la ALM que se centra principalmente en la etapa de desarrollo. Mientras que el SDLC solo abarca una aplicación, la ALM suele utilizarse de manera más amplia para gestionar toda una cartera de productos de software.

Un ciclo de vida de desarrollo de software (SDLC) es una metodología formal o informal para diseñar, crear y mantener software (incluido el código integrado en hardware).

Pocos modelos SDLC abordan explícitamente la seguridad del software en detalle, por lo que las prácticas de desarrollo de software seguro suelen integrarse en cada modelo SDLC. Independientemente del modelo SDLC utilizado, las prácticas de desarrollo de software seguro deben integrarse en él por tres razones: reducir la cantidad de vulnerabilidades en el software publicado, reducir el impacto potencial de la explotación de vulnerabilidades no detectadas o no abordadas, y abordar las causas raíz de las vulnerabilidades para evitar que vuelvan a ocurrir. Las vulnerabilidades incluyen no solo errores causados por fallos de codificación, sino también debilidades causadas por configuraciones de seguridad, suposiciones de confianza incorrectas y análisis de riesgos obsoletos.

La mayoría de los aspectos de seguridad pueden abordarse varias veces dentro de un ciclo de vida del desarrollo (SDLC), pero, en general, cuanto antes se aborde la seguridad, menor será el esfuerzo y el coste necesarios para lograr el mismo nivel de seguridad. Este principio, conocido como desplazamiento a la izquierda, es fundamental independientemente del modelo SDLC. El desplazamiento a la izquierda minimiza la deuda técnica que requeriría la corrección de fallos de seguridad iniciales en etapas avanzadas del desarrollo o una vez que el software esté en producción. Además, el desplazamiento a la izquierda puede resultar en un software con mayor seguridad y resiliencia.

2.2.2. Tipos de modelo SDLC

- **Modelo en Cascada**: Es el enfoque lineal clásico. Cada fase (Requisitos, Diseño, Implementación, Verificación, Mantenimiento) debe completarse antes de pasar a la siguiente. (Lewis)

- Ventaja: Fácil de gestionar y documentar.
- Desventaja: Muy rígido; si el cliente cambia de opinión al final, el costo es altísimo.

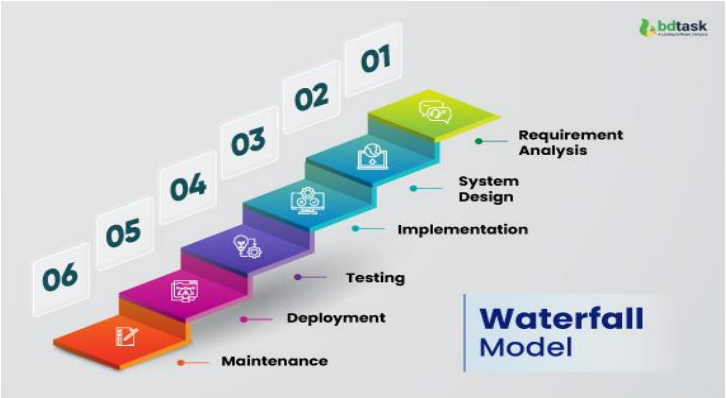


Figura 3 Modelo cascada (bdtask, s.f.)

- **Modelo en V (Validación y Verificación):** Una extensión de la cascada donde cada fase de desarrollo tiene una fase de prueba correspondiente. Se enfoca intensamente en el control de calidad desde el inicio. (Forsberg, K., & Mooz, H., 1991)

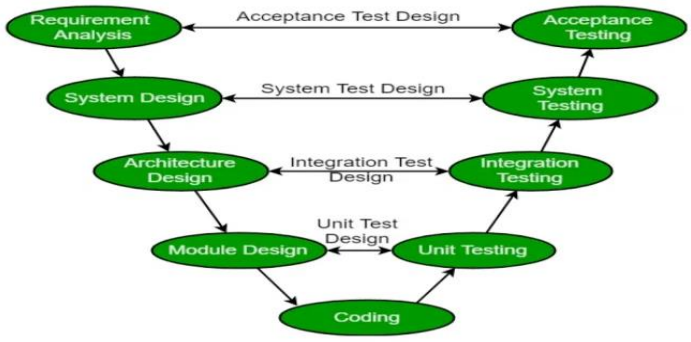


Figura 4 Modelo en V (geeksforgeeks, s.f.)

- **Modelo en Espiral:** El modelo combina los pequeños ciclos repetidos del modelo iterativo con el flujo secuencial y lineal del modelo de cascada para dar prioridad al análisis de riesgos. Puede usar el modelo de espiral para garantizar la actualización y mejora graduales del software mediante la creación de prototipos en cada fase.

  Ventajas y desventajas

  El modelo de espiral es adecuado para proyectos grandes y complejos que requieren cambios frecuentes. Sin embargo, puede ser costoso para proyectos pequeños con objetivos muy concretos. (Amazon, s.f.)

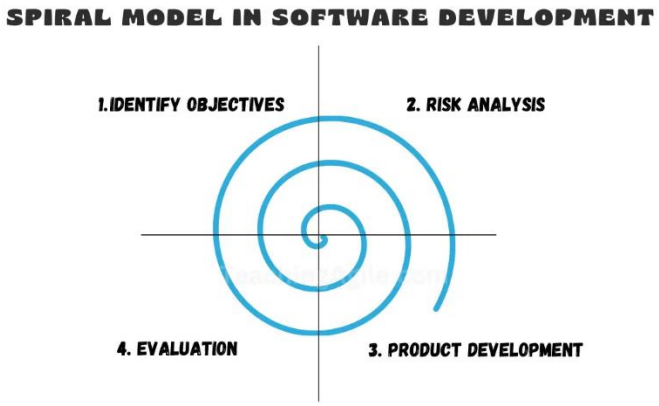


Figura 5 Modelo espiral (teachingagile, s.f.)

- **Agiles:** No es un modelo único, sino una filosofía (basada en el Manifiesto Ágil). Divide el proyecto en pequeñas entregas llamadas "Sprints".

  Marcos populares:

  - Scrum: Es un marco de administración que los equipos utilizan para organizarse por cuenta propia y trabajar en aras de alcanzar un objetivo común. Describe un conjunto de reuniones, herramientas y funciones para entregar proyectos de forma eficiente. Al igual que un equipo deportivo

que practica para un importante partido, las prácticas de Scrum permiten a los equipos de trabajo gestionarse por cuenta propia, aprender a partir de la experiencia y adaptarse al cambio. Los equipos de software utilizan Scrum para resolver problemas complejos de forma rentable y sostenible. (Amazon, s.f.)

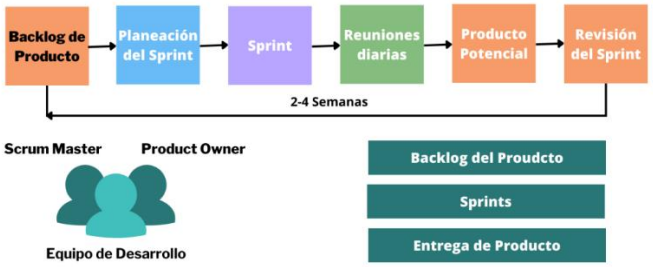


Figura 6 Flujo de trabajo en Scrum (productiviza, s.f.)

- XP El desarrollo de software requiere de gran concentración y de largas horas de trabajo, lo que hace que afrontar este tipo de trabajo sin ayuda pueda resultar una tarea de proporciones titánicas. Por suerte, existen recursos como la metodología XP (Extreme Programming) un método ágil que permite a estos profesionales acelerar su labor sin sacrificar por ello la calidad del producto final.

  Tener claro sobre qué principios se fundamenta la programación extrema puede transformar por completo su vida profesional, ¡por eso vamos a contarles qué es extreme programming y cómo pueden sacarle el máximo provecho posible. (godaddy, s.f.)

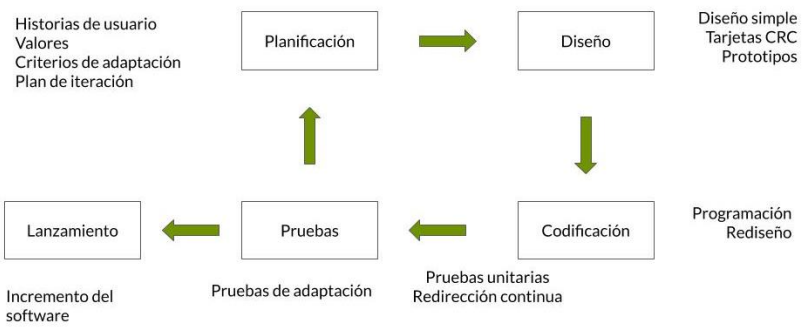


Figura 7 Metodología XP (sinnaps, s.f.)

- **DevOps:** Bajo un modelo de DevOps, los equipos de desarrollo y operaciones ya no están "aislados". A veces, los dos equipos se fusionan en uno solo, donde los ingenieros trabajan en todo el ciclo de vida de la aplicación, desde el desarrollo y las pruebas hasta la implementación y las operaciones, y desarrollan una variedad de habilidades no limitadas a una única función.

  En algunos modelos de DevOps, los equipos de control de calidad y de seguridad también se integran más con el desarrollo y las operaciones e intervienen durante todo el ciclo de vida de la aplicación. Cuando la seguridad es la prioridad de todos los miembros de DevOps, a veces se conoce como operaciones de seguridad de desarrollo.

  Los equipos utilizan prácticas para automatizar los procesos que anteriormente habían sido manuales y lentos. Utilizan una pila de tecnología y herramientas que los ayudan a operar y mejorar aplicaciones de forma rápida y confiable. Además, estas herramientas también ayudan a los ingenieros a realizar de forma independiente tareas que normalmente hubieran requerido la ayuda de otros equipos (por ejemplo, implementar código o aprovisionar infraestructura), incrementando así todavía más la velocidad del equipo. (aws.amazon, s.f.)

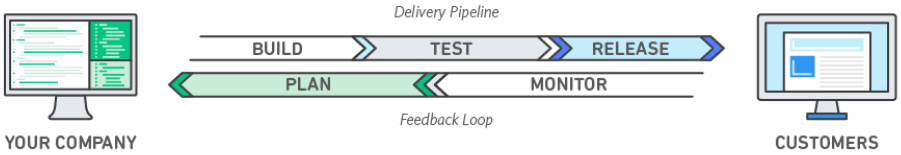


Figura 8 Modelo DevOps (aws.amazon, s.f.)

A continuación, se presenta un análisis de los modelos.

Modelo	Flexibilidad	Requisitos	Sugerido
Cascada	Muy Baja	Claros y fijos	Proyectos gubernamentales o de infraestructura rígida.
Espiral	Media	Complejos	Proyectos de alto riesgo y gran presupuesto.
Agile	Muy Alta	Cambiantes	Startups, aplicaciones web y software comercial móvil.
DevOps	Extrema	Continuos	Sistemas que requieren actualizaciones constantes (SaaS).

Tabla 2 Comparación de modelos SDLC Fuente Propia

2.2.3. La importancia de la Seguridad en el Ciclo de Desarrollo de Software (SDLC)

La integración de la seguridad desde las etapas iniciales del ciclo de desarrollo de software (SDLC) ha emergido como una necesidad crítica en el panorama digital actual. Tradicionalmente, la seguridad se consideraba una fase posterior al desarrollo, lo que a menudo resultaba en la identificación tardía de vulnerabilidades costosas de solucionar y potencialmente explotables por actores maliciosos (McGraw, 2006). El cambio hacia un enfoque de "seguridad por diseño" (security by design) reconoce que la seguridad debe ser una consideración inherente en cada fase del SDLC, desde la planificación y el diseño hasta la implementación, las pruebas, el despliegue y el mantenimiento.

Según (Amazon, s.f.) las ventajas del SDLC:

- Mayor visibilidad del proceso de desarrollo para todas las partes interesadas implicadas
- Una estimación, planificación y programación eficientes
- Mejoras en la administración de riesgos y estimación de costos
- Entregas de software sistemáticas y mayor satisfacción de los clientes

2.2.4. Amenazas y Vulnerabilidades en el SDLC

El SDLC es susceptible a una variedad de amenazas y la introducción de vulnerabilidades en cada una de sus etapas. Durante la fase de planificación y requisitos, una comprensión incompleta o ambigua de los requisitos de seguridad puede llevar a la omisión de funcionalidades de seguridad críticas. En la fase de diseño, decisiones arquitectónicas deficientes pueden introducir vulnerabilidades estructurales difíciles de mitigar posteriormente. La fase de implementación es donde se introducen la mayoría de las vulnerabilidades a través de errores de codificación, el uso de bibliotecas inseguras o la falta de validación de entradas. Durante las pruebas, si las pruebas de seguridad no son exhaustivas, las vulnerabilidades pueden pasar desapercibidas. Finalmente, en la fase de despliegue y mantenimiento, una configuración inadecuada o la falta de aplicación de parches de seguridad pueden crear puntos débiles explotables (Howard & LeBlanc, 2003).

2.2.5. Consecuencias de las Fallas de Seguridad en el Software

Las consecuencias de las fallas de seguridad en el software pueden ser significativas, especialmente para instituciones financieras como las cajas de ahorro. Estas consecuencias pueden incluir:

- Pérdidas financieras: Debido a fraudes, robo de información financiera o interrupción de servicios.

- Daño a la reputación: La pérdida de confianza de los clientes puede tener efectos a largo plazo en la sostenibilidad de la institución.
- Incumplimiento normativo: Las regulaciones financieras a menudo exigen altos estándares de seguridad, y las brechas pueden resultar en sanciones legales y financieras.
- Interrupción de operaciones: Los ataques pueden paralizar sistemas críticos, afectando la capacidad de la caja de ahorro para ofrecer servicios a sus miembros.
- Compromiso de datos sensibles: La exposición de información personal y financiera de los miembros puede tener graves implicaciones legales y de privacidad.

2.2.6. El Marco NIST de Seguridad del Desarrollo de Software (SSDF)

El Marco de Seguridad del Desarrollo de Software (SSDF) del Instituto Nacional de Estándares y Tecnología (NIST) es un conjunto de prácticas y recomendaciones diseñadas para ayudar a las organizaciones a reducir el riesgo de vulnerabilidades en el software. Publicado inicialmente en 2020, el SSDF proporciona una guía estructurada para integrar consideraciones de seguridad en todas las etapas del SDLC. Su enfoque se centra en las siguientes cuatro prácticas fundamentales:

2.2.6.1. Descripción General del SSDF

El SSDF no es un estándar prescriptivo único, sino más bien un marco flexible que las organizaciones pueden adaptar a sus necesidades y contextos específicos. Se basa en principios de seguridad probados y en las lecciones aprendidas de incidentes de seguridad pasados. El objetivo principal del SSDF es fomentar un cambio cultural hacia el desarrollo de software seguro, donde la seguridad sea una responsabilidad compartida por todos los involucrados en el SDLC.

Las Cuatro Prácticas Fundamentales del SSDF

El SSDF se organiza en torno a cuatro prácticas esenciales, cada una con una serie de tareas y subtareas detalladas:

- **Preparar la Organización (Prepare the Organization)**

  Esta práctica se centra en establecer el contexto organizacional y los prerrequisitos necesarios para un programa de desarrollo de software seguro. Incluye tareas como:

  - Definir políticas y procedimientos de seguridad del SDLC: Establecer directrices claras sobre cómo se debe abordar la seguridad en cada fase del desarrollo.
  - Definir roles y responsabilidades de seguridad: Asignar responsabilidades específicas a los miembros del equipo en relación con la seguridad del software.
  - Proporcionar capacitación y concientización en seguridad: Asegurar que todos los involucrados en el SDLC tengan el conocimiento y las habilidades necesarias para desarrollar software seguro.
  - Establecer un entorno de desarrollo seguro: Configurar la infraestructura y las herramientas de desarrollo de manera segura.

- **Proteger el Software (Protect the Software)**

  Esta práctica se enfoca en proteger la integridad y la confidencialidad del software durante todo su ciclo de vida. Incluye tareas como:

  - Gestionar el acceso al código fuente y otros activos: Implementar controles de acceso robustos para proteger la información sensible.
  - Proteger contra la manipulación no autorizada: Implementar mecanismos para detectar y prevenir modificaciones no autorizadas del código.

- Mantener la integridad del entorno de desarrollo: Asegurar que el entorno de desarrollo no esté comprometido.
- Prepararse para la recuperación ante desastres: Establecer planes para la continuidad del desarrollo en caso de incidentes.

- **Producir Software Bien Asegurado (Produce Well-Secured Software)**

Esta práctica aborda las actividades específicas de seguridad que deben integrarse en el proceso de desarrollo. Incluye tareas como:

- Diseñar para la seguridad: Incorporar consideraciones de seguridad en la arquitectura y el diseño del software.
- Implementar prácticas de codificación segura: Seguir directrices y estándares de codificación para evitar vulnerabilidades comunes.
- Realizar pruebas de seguridad: Integrar pruebas estáticas, dinámicas y de penetración para identificar vulnerabilidades.
- Gestionar vulnerabilidades de terceros: Evaluar y mitigar los riesgos asociados con el uso de componentes de software de terceros.
- Generar una lista de materiales de software (SBOM): Mantener un inventario de los componentes de software utilizados.

- **Responder a las Vulnerabilidades**

Esta práctica se centra en las actividades necesarias para identificar, abordar y mitigar las vulnerabilidades que se descubren en el software después de su despliegue. Incluye tareas como:

- Establecer un proceso de gestión de vulnerabilidades: Definir cómo se identificarán, analizarán, priorizarán y solucionarán las vulnerabilidades.

- Comunicar información sobre vulnerabilidades: Informar a las partes interesadas relevantes sobre las vulnerabilidades descubiertas y las acciones tomadas.
- Analizar las causas raíz de las vulnerabilidades: Comprender por qué se introdujeron las vulnerabilidades para prevenir futuras ocurrencias.
- Aplicar parches y actualizaciones de seguridad: Implementar las correcciones necesarias de manera oportuna.

2.2.6.2. Relación del SSDF con Otros Marcos y Estándares de Seguridad (OWASP, ISO/IEC 27034)

La selección del marco NIST SSDF para esta investigación se fundamentó en que proporciona un enfoque práctico y adaptable orientado específicamente al desarrollo seguro de software, permitiendo integrar controles de seguridad directamente dentro del ciclo de vida del desarrollo.

A diferencia de modelos como ISO/IEC 27034, que requieren estructuras complejas de gobernanza y altos niveles de madurez organizacional, NIST SSDF presenta una implementación más flexible y adecuada para organizaciones con recursos limitados, como las cajas de ahorro pertenecientes a REFIDER.

Asimismo, OWASP SAMM se orienta principalmente a la evaluación de madurez organizacional en desarrollo seguro, mientras que NIST SSDF facilita la incorporación directa de prácticas técnicas aplicables al diseño e implementación de un prototipo financiero seguro.

En el contexto de REFIDER, donde inicialmente no existía un SDLC formal y solo se contaba con un responsable de procesos tecnológicos, la adopción de un marco práctico y escalable como NIST SSDF permitió implementar controles preventivos desde la arquitectura, fortalecer la

seguridad del prototipo financiero, reducir vulnerabilidades y mejorar progresivamente el nivel de madurez de seguridad.

Por esta razón, NIST SSDF fue considerado el marco más adecuado para la presente investigación, debido a su equilibrio entre aplicabilidad técnica, flexibilidad metodológica y alineación con entornos financieros cooperativistas de pequeña escala.

2.2.7. Marco Legal y Normativo Aplicable

El desarrollo seguro de software en instituciones financieras cooperativistas no solo debe alinearse con estándares técnicos internacionales como National Institute of Standards and Technology SSDF, sino también cumplir con disposiciones legales y regulatorias emitidas por organismos nacionales de control financiero y protección de datos.

En el contexto ecuatoriano, las cajas de ahorro y cooperativas que pertenecen al sector de la Economía Popular y Solidaria se encuentran reguladas principalmente por la Superintendencia de Economía Popular y Solidaria (SEPS), la cual establece lineamientos relacionados con seguridad de la información, continuidad operativa, gestión tecnológica, administración de riesgos y protección de canales electrónicos.

Entre las principales normativas consideradas se encuentra la Resolución No. SEPS-IGS-IGT-IGJ-INGINT-INTIC-INSESF-INR-DNSI-2022-002 emitida por la Superintendencia de Economía Popular y Solidaria, la cual establece lineamientos relacionados con gestión del riesgo operativo, seguridad de la información, control de accesos, monitoreo de eventos y continuidad operativa en entidades financieras del sector popular y solidario. Asimismo, se tomó como referencia la Resolución No. SEPS-IGT-IGS-INSESF-INR-INGINT-INSEPS-009, orientada al fortalecimiento de la administración tecnológica y control operativo de plataformas financieras.

De igual manera, se consideró la Norma de Control respecto a la Seguridad Física y Electrónica, la cual establece mecanismos para proteger infraestructuras tecnológicas, sistemas informáticos y canales electrónicos frente a accesos no autorizados, incidentes de seguridad y fraudes tecnológicos.

En el ámbito legal, la investigación incorporó criterios establecidos en la Asamblea Nacional del Ecuador (LOPDP), orientados a garantizar la confidencialidad, integridad y protección de los datos financieros procesados por el sistema. Además, se consideraron principios establecidos en la Asamblea Nacional del Ecuador (LOTAIP), especialmente en aspectos relacionados con trazabilidad, auditoría y control documental. Finalmente, la Asamblea Nacional del Ecuador aportó lineamientos relacionados con modernización tecnológica, ciberseguridad y fortalecimiento de servicios digitales.

Estas normativas guardan relación directa con las prácticas del marco NIST SSDF implementadas en la investigación, particularmente en aspectos asociados con autenticación segura, control de accesos, validación de entradas, auditoría de eventos, protección de datos, monitoreo de vulnerabilidades y gestión segura del ciclo de vida del software. De esta manera, el prototipo financiero desarrollado para REFIDER no solo se alineó con estándares internacionales de desarrollo seguro, sino también con el marco regulatorio ecuatoriano aplicable al sector financiero cooperativista.

CAPITULO III

3. MARCO METODOLOGICO

La seguridad informática está regulada por normas que permiten implementar mecanismos de seguridad para cuidar los activos relacionados con los sistemas informáticos. Una de las encargadas estandarizar y regular es la NIST, donde se describen los lineamientos de seguridad para el desarrollo de software seguro para organizaciones y de esta manera garantizar la confidencialidad, autenticidad e integridad de la información.

Importancia de la Seguridad en el SDLC: Integrar la seguridad en cada fase del SDLC, conocido como Secure SDLC, es fundamental en el entorno actual donde las amenazas cibernéticas son sofisticadas y los sistemas de software manejan información sensible.

3.1. Descripción del área de estudio / Descripción del grupo de estudio

La Red de Desarrollo Rural (REFIDER) se encuentra ubicada en la ciudad de Ibarra, parroquia El Sagrario, provincia de Imbabura. REFIDER constituye una red de apoyo para entidades financieras rurales del norte del Ecuador, brindando servicios relacionados con contabilidad, auditoría, tributación y sistemas informáticos.

REFIDER está conformada por 12 entidades socias, las cuales utilizan soluciones tecnológicas para la gestión de procesos financieros y administrativos. Dentro de su estructura organizacional, la gestión tecnológica se encuentra centralizada en un único responsable del área de Tecnologías de la Información (TI), quien cumple funciones relacionadas con soporte técnico, administración de sistemas, mantenimiento, desarrollo de software y gestión de base de datos.

Actualmente, REFIDER no dispone de un modelo formal de ciclo de vida de desarrollo de software (SDLC), ni de políticas estructuradas de seguridad aplicadas al desarrollo de aplicaciones.

Las actividades de desarrollo y mantenimiento se realizan de manera empírica y reactiva, dependiendo del conocimiento técnico del responsable de TI.

En este contexto, la investigación se enfocó en evaluar la aplicación del marco NIST Secure Software Development Framework (SSDF) en el desarrollo de un módulo financiero de caja, correspondiente a operaciones de depósitos y retiros, desarrollado como caso de estudio para analizar la incorporación de prácticas de seguridad en el proceso de desarrollo de software.

La población estuvo integrada por:

Grupo	Cantidad
Cajas de ahorro socias de REFIDER	12
Responsable del área TI	1
Total población objeto de estudio	**13**

Tabla 3 Población de estudio Fuente Propia

Debido a las características de la organización y al tamaño reducido de la población, se aplicó un muestreo no probabilístico por conveniencia, considerando la totalidad de los elementos involucrados en el estudio.

La unidad de análisis estuvo conformada por:

- Los procesos de desarrollo de software existentes en REFIDER
- El módulo de caja desarrollado como caso de estudio
- Las prácticas de seguridad aplicadas durante el desarrollo

3.2. Enfoque y tipo de investigación

La presente investigación se desarrolló bajo un enfoque mixto, debido a que el estudio requirió integrar métodos cualitativos y cuantitativos para analizar la seguridad del proceso de desarrollo de software en REFIDER y evaluar la aplicación del marco NIST Secure Software Development Framework (SSDF).

La naturaleza compleja de la seguridad en el desarrollo de software demanda comprender tanto los aspectos relacionados con procesos, prácticas y percepciones organizacionales, como los resultados medibles derivados de la implementación de controles de seguridad. En este contexto, el enfoque mixto permitió combinar la profundidad interpretativa de los datos cualitativos con la capacidad de medición y validación de los datos cuantitativos.

La investigación adoptó específicamente un diseño mixto de tipo Exploratorio Secuencial, debido a que inicialmente se realizó una fase cualitativa orientada al diagnóstico y comprensión del entorno de desarrollo de software existente en REFIDER, caracterizado por la ausencia de un ciclo de vida de desarrollo de software (SDLC) formalmente definido.

3.3.Procedimientos de investigación

La investigación se desarrolló mediante un procedimiento estructurado en tres fases metodológicas, alineadas con los objetivos específicos planteados y orientadas a evaluar la aplicación del marco NIST Secure Software Development Framework (SSDF) en el desarrollo de software de REFIDER.

FASE 1: Diagnóstico y adaptación del marco NIST SSDF

Esta fase tiene como objetivo analizar el estado actual del desarrollo de software en REFIDER e identificar las necesidades de seguridad existentes, considerando que la organización no dispone de un modelo formal de ciclo de vida de desarrollo de software (SDLC).

Durante esta fase se realizaron las siguientes actividades:

Revisión de literatura

Se efectuó una revisión bibliográfica y documental relacionada con:

- NIST Secure Software Development Framework (SSDF)
- Seguridad en el ciclo de desarrollo de software
- OWASP SAMM
- BSIMM
- Seguridad en sistemas financieros
- Buenas prácticas de desarrollo seguro

Asimismo, se analizaron investigaciones previas y documentación técnica relacionada con seguridad del software y marcos de desarrollo seguro.

Recolección de datos inicial

Se recopiló información institucional de REFIDER y de las cajas de ahorro afiliadas, identificando:

- Procesos actuales de desarrollo
- Prácticas de seguridad existentes
- Herramientas utilizadas
- Debilidades en el desarrollo de software

Aplicación de instrumentos

Se aplicaron técnicas de investigación como:

- Encuestas
- Observación directa
- Análisis documental

Evaluación inicial mediante SSDF

Se utilizó una matriz de evaluación basada en las prácticas del NIST SSDF:

PO (Prepare the Organization)

PS (Protect the Software)

PW (Produce Well-Secured Software)

RV (Respond to Vulnerabilities)

La escala de madurez utilizada en la investigación fue adaptada tomando como referencia modelos reconocidos de evaluación de capacidades y seguridad, tales como Capability Maturity Model Integration (CMMI) y OWASP Software Assurance Maturity Model (SAMM). Estos modelos proporcionan estructuras progresivas para medir el nivel de formalización, gestión y optimización de procesos relacionados con el desarrollo seguro de software. La adaptación de dichos enfoques permitió cuantificar el grado de implementación de las prácticas definidas en el marco NIST SSDF.

Nivel	Valoración	Descripción
0	No existe	No se identificó evidencia de la práctica evaluada.
1	Muy bajo	La práctica existe de manera informal, reactiva o empírica.
2	Bajo	Existen controles básicos, pero sin formalización ni seguimiento.
3	Medio	La práctica se encuentra parcialmente documentada e implementada.
4	Alto	La práctica está formalizada, controlada y aplicada consistentemente.
5	Óptimo	La práctica se encuentra integrada, monitoreada y en mejora continua.

Tabla 4 Valoraciones

Para el procesamiento y análisis de la información recopilada durante la investigación se utilizaron técnicas de análisis cualitativo y cuantitativo, permitiendo interpretar los resultados obtenidos en relación con la aplicación del marco NIST Secure Software Development Framework (SSDF).

El análisis cualitativo se realizó mediante categorización temática y análisis interpretativo de la información recopilada. Adicionalmente, se consideró el uso de herramientas como Atlas.ti para la organización de datos cualitativos.

El análisis cuantitativo se aplicó a los datos obtenidos mediante encuestas y matrices de evaluación SSDF. Los resultados fueron organizados y procesados utilizando herramientas tecnológicas como, Microsoft Excel para hacer la tabulación y gráficos.

Para determinar el nivel de madurez global del desarrollo seguro en REFIDER, se utilizó un promedio aritmético de los resultados obtenidos en los cuatro dominios del marco NIST SSDF.

El nivel de madurez global se calculó mediante la siguiente fórmula:

NMG= (PO+PS+PW+RV) / 4

FASE 2: Desarrollo e implementación del módulo financiero

Como parte de la aplicación del marco NIST Secure Software Development Framework (SSDF), se incorporó el enfoque "Shift Left Security", el cual propone integrar prácticas de seguridad desde las primeras etapas del ciclo de desarrollo de software, en lugar de tratarlas únicamente en fases finales de validación o despliegue.

Debido a que REFIDER no dispone de un ciclo de vida de desarrollo de software (SDLC) formalmente definido, en la presente investigación se diseñó un ciclo de vida seguro adaptado al desarrollo del módulo financiero de caja, correspondiente a operaciones de depósitos y retiros.

El diseño metodológico permitió insertar controles de seguridad en cada etapa del desarrollo del prototipo, alineando las actividades del proyecto con las prácticas propuestas por el NIST SSDF.

Esta fase tuvo como objetivo implementar un módulo de caja para operaciones de depósitos y retiros, aplicando prácticas de seguridad basadas en el marco NIST SSDF.

Se definieron los requerimientos funcionales y de seguridad del módulo financiero, considerando:

- Autenticación
- Control de acceso
- Validación de datos
- Auditoría de eventos

FASE 3: Validación y evaluación del módulo desarrollado

Esta fase tuvo como objetivo validar el impacto de la aplicación del NIST SSDF en la seguridad del módulo desarrollado. Con el propósito de garantizar la validez de contenido de los instrumentos utilizados en la investigación, se aplicó la técnica de juicio de expertos.

Se evaluaron indicadores relacionados con:

- Reducción de vulnerabilidades
- Control de acceso
- Trazabilidad
- Validación de datos
- cumplimiento de seguridad

Los datos cualitativos obtenidos fueron organizados y analizados para identificar patrones y problemáticas relacionadas con la seguridad del desarrollo.

Por otra parte, los datos cuantitativos fueron procesados mediante tablas comparativas y métricas descriptivas, permitiendo evaluar el nivel de mejora alcanzado.

3.4. Consideraciones bioéticas

En esta investigación se realizaron entrevistas al personal de la Dirección de Tecnología de REFIDER. Como parte del protocolo se puso en conocimiento de los jefes inmediatos el estudio que se va a realizar. Posteriormente se informó a los participantes sobre la privacidad y confiabilidad que se va a tener sobre el uso que se va a dar a la información suministrada, acatando los lineamientos que se tiene sobre la ley de protección de datos personales para que la misma no sea divulgada y peor usada de manera anti técnica. Durante todo el proceso de investigación se consideraron aspectos éticos y normativos.

CAPITULO IV

4. RESULTADOS

4.1. Evaluación del estado actual del SDLC en las cajas de ahorro de REFIDER

A partir de la aplicación de los instrumentos definidos en el marco metodológico (análisis documental y matrices de evaluación), se identificó que las cajas de ahorro socias de REFIDER presentan un nivel bajo de madurez en cuanto a la integración de prácticas de seguridad dentro del ciclo de vida de desarrollo de software (SDLC).

Los principales hallazgos evidencian que:

- Ausencia de integración de seguridad en fases tempranas del desarrollo.
- Falta de políticas formales de desarrollo seguro.
- No tienen un ciclo de vida de desarrollo de software.
- Carencia de procesos estructurados de gestión de vulnerabilidades

Estos resultados coinciden con lo expuesto por McGraw (2006), quien señala que la seguridad tratada de forma reactiva incrementa la probabilidad de vulnerabilidades críticas.

Esta situación motivó la implementación de un módulo de caja (depósitos y retiros) como caso de estudio, en el cual se aplicaron las prácticas del marco NIST SSDF, con el objetivo de evaluar su impacto en la seguridad del software.

Se encontraron problemas como:

Desarrollo sin metodología, no se sigue ningún modelo como cascada, ágil o DevSecOps.

Código sin validación, entradas de usuario no controladas.

Acceso directo a base de datos uso de consultas SQL sin protección.

Falta de control de acceso permisos no estructurados.

Ausencia de auditoría, no se registran eventos críticos.

ID	Vulnerabilidad	Impacto	Severidad
V1	Inyección SQL	Alto	Crítico
V2	Falta de validación	Alto	Alto
V3	Control de acceso débil	Alto	Alto
V4	Falta de auditoría	Medio	Medio
V5	Manejo incorrecto de errores	Medio	Medio

Tabla 5 Vulnerabilidades encontradas Fuente Propia

4.2. Nivel de alineación con el marco NIST SSDF

El análisis inicial evidenció que REFIDER no cuenta con un modelo estructurado de desarrollo de software, lo cual implica que las actividades de análisis, diseño, desarrollo y pruebas se realizan de manera empírica.

Principales hallazgos:

- No existe documentación de procesos de desarrollo
- No se definen requisitos de seguridad
- No hay separación formal de ambientes (desarrollo / pruebas / producción)
- No se realizan pruebas de seguridad
- No existe control de versiones estructurado
- No se implementan estándares de codificación

La inexistencia de un SDLC genera:

- Desarrollo inconsistente
- Alta probabilidad de errores
- Vulnerabilidades no controladas

- Dependencia del conocimiento individual



Figura 9 Integración del SSDF en ciclo de desarrollo (SDLC) Fuente Propia

Se aplicó una matriz de evaluación como base para obtener la información inicial, para esto se realizó una encuesta al responsable de TI de REFIDER teniendo en cuenta las cuatro prácticas del NIST SSDF:

- Preparar la organización (PO)
- Proteger el software (PS)
- Producir software bien asegurado (PW)
- Responder a vulnerabilidades (RV)

Práctica	Código	Criterio	Descripción	Nivel (0-5)	Evidencia identificada	Justificación de la valoración
Preparar Organización	PO.1	Políticas	Existencia de políticas de desarrollo seguro	0	No se encontraron manuales, políticas ni procedimientos documentados	Se asignó nivel 0 debido a la inexistencia de documentación formal relacionada con desarrollo seguro

	PO.2	Roles	Definición de roles de seguridad	0	No existe definición formal de responsabilidades de seguridad	El responsable TI ejecuta múltiples funciones sin separación de roles
	PO.3	Capacitación	Formación en seguridad	1	El responsable TI indicó conocimientos empíricos en seguridad	Existen conocimientos básicos, pero no capacitación formal ni certificaciones
	PO.4	Entorno seguro	Configuración segura de desarrollo	1	Se observó uso básico de credenciales y restricciones locales	Existen controles mínimos sin estandarización formal
Proteger Software	PS.1	Control de acceso	Acceso a código fuente	2	El acceso al código fuente es restringido únicamente al responsable TI	Existe control de acceso empírico, aunque no documentado ni gestionado formalmente
	PS.2	Integridad	Protección contra modificaciones	1	No existe control de versiones formal ni trazabilidad	Se identificaron respaldos manuales sin mecanismos de integridad
	PS.3	Backup	Respaldo y recuperación	1	Existen copias manuales de base de datos	Los respaldos son realizados ocasionalmente sin políticas establecidas
	PS.4	Seguridad infra	Seguridad del entorno	1	Uso básico de antivirus y credenciales locales	No existen controles avanzados de seguridad de infraestructura

Producir Software Seguro	PW.1	Diseño seguro	Arquitectura segura	1	El diseño se realiza empíricamente	No se utilizan modelos formales de arquitectura segura
	PW.2	Codificación	Prácticas seguras	1	Validaciones básicas implementadas manualmente	No existe estándar formal de codificación segura
	PW.3	Pruebas	SAST/DAST	0	No se evidenció uso de herramientas de pruebas de seguridad	No existen análisis estáticos ni dinámicos de vulnerabilidades
	PW.4	Dependencias	Gestión de librerías	1	Uso de librerías sin control formal de versiones	No existe monitoreo de vulnerabilidades en dependencias
	PW.5	SBOM	Inventario de componentes	1	Se identificó conocimiento parcial de componentes utilizados	No existe inventario documentado de componentes de software
Respuesta Vulnerabilidades	RV.1	Gestión	Proceso de vulnerabilidades	1	Correcciones realizadas de manera reactiva	No existe procedimiento formal de gestión de vulnerabilidades
	RV.2	Comunicación	Reporte de incidentes	1	Reportes informales mediante comunicación verbal	No existe mecanismo formal de notificación o escalamiento
	RV.3	Análisis raíz	Mejora continua	2	El responsable TI realiza revisiones posteriores a incidentes	Existe análisis empírico de errores, aunque no documentado

	RV.4	Parches	Actualizaciones	1	Actualizaciones manuales del sistema	No existe planificación formal de gestión de parches

Tabla 6 Matriz de evaluación inicial fuente propia

Escala:

- 0 = No existe
- 1 = Muy bajo
- 2 = Bajo
- 3 = Medio
- 4 = Alto
- 5 = Óptimo

Los resultados obtenidos evidencian un bajo nivel de madurez en seguridad:

Dimensión SSDF	Promedio
PO	0.5
PS	1.25
PW	0.8
RV	1.25

Tabla 7 Promedio de evaluación Fuente Propia

Práctica SSDF	Nivel de cumplimiento	Observaciones
PO	Bajo	No existen políticas ni capacitación formal
PS	Bajo	Controles de acceso limitados
PW	Bajo	No se aplican pruebas de seguridad
RV	Bajo	No existe gestión de vulnerabilidades óptimas

Tabla 8 Nivel de cumplimiento Fuente Propia

El análisis evidencia que:

- La práctica más débil es Preparar Organización (PO).
- Existe una brecha significativa en producción de software seguro (PW).

- La seguridad no está integrada de forma transversal en el SDLC.

El nivel global de madurez obtenido fue de 0,95/5, clasificándose como nivel bajo con esto se confirma que REFIDER opera sin prácticas formales de desarrollo seguro.

El nivel global de madurez obtenido fue de 0,95/5, clasificándose como nivel bajo. Un valor inferior a 1.00 indica que las cajas de ahorro de REFIDER operan en un estado de seguridad predominantemente reactiva y dependiente del factor humano, donde la ausencia total de automatización en la verificación de código eleva la probabilidad de que fallas críticas pasen a producción de forma inadvertida. Se evidencia una brecha significativa entre las prácticas actuales y las recomendaciones del NIST SSDF (NIST, 2020).

4.3. Controles de seguridad implementados en el módulo financiero

Con el objetivo de fortalecer la seguridad del módulo financiero desarrollado para las cajas de ahorro pertenecientes a REFIDER, se implementaron diversos controles de seguridad alineados con las prácticas del marco National Institute of Standards and Technology Secure Software Development Framework (SSDF).

Los controles incorporados fueron seleccionados considerando, las vulnerabilidades identificadas durante el diagnóstico inicial, los riesgos asociados al manejo de información financiera y las recomendaciones establecidas por estándares y buenas prácticas de desarrollo seguro NIST SSDF.

La implementación de estos mecanismos permitió integrar seguridad en distintas etapas del ciclo de vida del software, aplicando el enfoque "Shift Left", el cual promueve incorporar controles preventivos desde fases tempranas del desarrollo.

Control	Implementación	Propósito
Autenticación	JWT	Verificar identidad de usuarios
Autorización	Roles	Restringir acceso según privilegios
Cifrado	BCrypt	Proteger contraseñas almacenadas
Validación	Bean Validation	Validar entradas y prevenir errores
Auditoría	Tabla auditoría	Registrar eventos y actividades
Integridad	Transacciones ACID	Garantizar consistencia financiera

Tabla 9 Controles de seguridad implementados Fuente Propia

4.3.1. Descripción de los controles implementados

- **Autenticación mediante JWT**

La autenticación del sistema fue implementada utilizando JSON Web Tokens (JWT), permitiendo validar la identidad de los usuarios de manera segura y sin mantener sesiones persistentes en el servidor.

El uso de JWT permitió:

- Proteger el acceso a endpoints
- Reducir riesgos de secuestro de sesión
- Controlar el acceso a operaciones sensibles.

- **Autorización basada en roles**

Se implementó un esquema de autorización basado en roles para restringir el acceso a funcionalidades específicas del sistema financiero.

Este mecanismo permitió definir privilegios diferenciados para los administradores, operadores y usuarios del sistema.

La implementación de roles fortalece el principio de mínimo privilegio y reduce riesgos de acceso no autorizado.

- **Cifrado de contraseñas mediante BCrypt**

Las credenciales de los usuarios fueron protegidas mediante el algoritmo BCrypt, utilizado para cifrar contraseñas antes de almacenarlas en PostgreSQL.

Esta implementación evita que el almacenamiento de contraseñas sea en texto plano, exposición de credenciales y ataques de recuperación de claves.

El cifrado de contraseñas constituye una práctica esencial de desarrollo seguro recomendada por NIST.

- **Validación de entradas mediante Bean Validation**

La validación de datos fue implementada utilizando anotaciones Bean Validation de Spring Boot, permitiendo verificar automáticamente los campos obligatorios, rangos válidos y tipos de datos.

Esto permitió prevenir la manipulación de parámetros, errores lógicos e ingreso de datos inconsistentes.

- **Auditoría y trazabilidad de eventos**

Se desarrolló un mecanismo de auditoría encargado de registrar actividades críticas realizadas dentro del módulo financiero.

Los registros almacenados incluyen a los usuarios, la acción realizada, la fecha y operación ejecutada.

De esta manera la auditoría permite monitorear actividades, detectar anomalías y fortalecer la trazabilidad operativa.

- **Integridad mediante transacciones ACID**

Las operaciones financieras fueron implementadas utilizando control transaccional ACID mediante Spring Boot y PostgreSQL.

Esto garantiza que las operaciones se ejecuten de forma atómica, los datos permanezcan consistentes y los errores no generen corrupción de información financiera.

4.4. Implementación del módulo financiero seguro basado en SSDF (prototipo)

El módulo financiero prototipo fue desarrollado aplicando controles de seguridad alineados con los cuatro dominios del marco National Institute of Standards and Technology Secure Software Development Framework (SSDF).

La implementación se realizó bajo el enfoque “Shift Left Security”, integrando controles de seguridad desde las primeras etapas del ciclo de vida del software.

4.4.1 Preparar la Organización (PO)

En esta fase se definieron controles organizacionales y de preparación del entorno de desarrollo seguro.

Código	Control implementado	Acción realizada
PO.1	Políticas básicas	Definición de lineamientos mínimos de desarrollo seguro
PO.2	Roles y responsabilidades	Restricción de acceso al código y funciones administrativas
PO.3	Capacitación	Investigación y aplicación de prácticas OWASP y SSDF
PO.4	Entorno seguro	Configuración segura de credenciales y entorno local

Tabla 10 Control Implementado PO Fuente Propia

Evidencias implementadas

- Definición de usuarios de acceso y seguridades
- Restricciones de credenciales
- Separación lógica de funciones

Mejora obtenida

Antes de la investigación, REFIDER no disponía de:

- Políticas de desarrollo seguro
- Controles organizacionales
- Prácticas formales de seguridad

Con la implementación del prototipo se incorporaron controles alineados al dominio PO del SSDF.

4.4.2 Proteger el Software (PS)

Esta fase se enfocó en implementar controles orientados a proteger el acceso, integridad y seguridad del software y de la información financiera.

Código	Control implementado	Acción realizada
PS.1	Control de acceso	Implementación de autenticación JWT
PS.2	Integridad	Firma segura de tokens mediante HS512
PS.3	Backup	Respaldo lógico de base de datos PostgreSQL
PS.4	Seguridad infraestructura	Configuración segura mediante Spring Security

Tabla 11 Controles Implementados PS Fuente Propia

Implementación técnica

```java
@Configuration
@EnableWebSecurity
public class SecurityConfig {

    @Bean
    public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
        http
            .csrf().disable()
            .sessionManagement()
            .sessionCreationPolicy(SessionCreationPolicy.STATELESS)
            .and()
            .authorizeHttpRequests()
            .requestMatchers("/auth/**").permitAll()
            .anyRequest().authenticated();

        return http.build();
    }
}
```

Figura 10 Configuración de seguridad mediante Spring Security Fuente Propia

La configuración de seguridad implementada mediante Spring Security permitió establecer mecanismos de autenticación y control de acceso sobre el módulo financiero desarrollado. Se aplicó una política de sesiones stateless utilizando JSON Web Tokens (JWT), restringiendo el acceso a los servicios protegidos únicamente a usuarios autenticados. Esta implementación se alinea con las prácticas PS.1 y PS.2 del marco NIST SSDF relacionadas con protección del software y control de acceso.

Configuración JWT	Descripción
setSubject(username)	Identifica al usuario
signWith(HS512, secret)	Firma criptográfica
setExpiration()	Tiempo de expiración
compact()	Generación final del token

Tabla 12 Implementación de autenticación JWT.

```
public String generarToken(String username) {
    return Jwts.builder()
        .setSubject(username)
        .setIssuedAt(new Date())
        .setExpiration(new Date(System.currentTimeMillis() + 86400000))
        .signWith(SignatureAlgorithm.HS512, secret)
        .compact();
}
```

Figura 11 Implementación de autenticación JWT

La autenticación del sistema fue implementada mediante JSON Web Tokens (JWT), permitiendo validar la identidad de los usuarios de forma segura y sin mantener sesiones persistentes en el servidor. Los tokens fueron firmados utilizando el algoritmo criptográfico HS512 y una clave secreta privada, garantizando la integridad de la información transmitida. Esta implementación fortalece el control de acceso y reduce riesgos asociados a secuestro de sesión y manipulación de credenciales.

4.4.3 Producir Software Bien Asegurado (PW)

En esta fase se implementaron prácticas de codificación segura y validación de entradas durante el desarrollo del módulo financiero.

Código	Control implementado	Acción realizada
PW.1	Diseño seguro	Arquitectura multicapa con separación lógica
PW.2	Codificación segura	Validaciones mediante DTO y anotaciones
PW.3	Pruebas	Validaciones funcionales y revisión manual
PW.4	Dependencias	Gestión de librerías mediante Maven
PW.5	Validación de entradas	Restricción de montos y campos obligatorios

Tabla 13 Controles implementados PW Fuente Propia

Evidencias implementadas

- Uso de @NotNull
- Uso de @DecimalMin
- Utilización de BigDecimal

- Consultas parametrizadas
- Gestión de dependencias con Maven

```java
@Transactional
public void realizarDeposito(Long cuentaId, BigDecimal monto) {

    Cuenta cuenta = cuentaRepository.findById(cuentaId)
        .orElseThrow(() -> new RuntimeException("Cuenta no encontrada"));

    cuenta.setSaldo(cuenta.getSaldo().add(monto));

    cuentaRepository.save(cuenta);

    auditoriaService.registrar("usuario", "Depósito realizado");
}
```

Figura 12 Gestión de Transacciones seguras Fuente Propia

Esta figura corresponde a la implementación del manejo seguro de transacciones financieras dentro del módulo de caja desarrollado para REFIDER.

La funcionalidad fue implementada utilizando control transaccional de Spring Boot y PostgreSQL, garantizando la integridad y consistencia de las operaciones financieras relacionadas con los depósitos, retiros y actualización de saldos

La implementación de transacciones seguras permitió garantizar la integridad de las operaciones financieras ejecutadas en el módulo de caja. Mediante el uso de la anotación @Transactional, el sistema asegura que todas las operaciones relacionadas con depósitos y retiros se ejecuten de forma atómica, evitando inconsistencias en la base de datos ante fallos o interrupciones del proceso. Esta práctica se alinea con los principios PW.1 y PW.2 del marco NIST SSDF relacionados con diseño y desarrollo seguro del software.

```java
public class TransaccionDTO {

    @NotNull
    private Long cuentaId;

    @NotNull
    @DecimalMin("0.01")
    private BigDecimal monto;

}
```

Figura 13 Validación de entradas mediante DTO Fuente Propia

Uno de los principios fundamentales del marco National Institute of Standards and Technology SSDF es el enfoque "Shift Left Security", el cual establece que los controles de seguridad deben incorporarse desde las primeras etapas del ciclo de vida del software y no únicamente al finalizar el desarrollo.

En el módulo financiero prototipo desarrollado para REFIDER, la seguridad fue considerada desde la arquitectura de la aplicación, integrando mecanismos preventivos en las diferentes capas del sistema como, la capa de presentación, la capa de negocio, la capa de persistencia y la capa de acceso a datos.

Esta estrategia permitió reducir la superficie de ataque y minimizar vulnerabilidades antes de que los datos lleguen a la lógica de negocio o a la base de datos PostgreSQL.

Uno de los principales mecanismos implementados fue la validación de entradas utilizando DTOs (Data Transfer Objects) junto con anotaciones Bean Validation de Spring Boot.

La validación fue aplicada directamente en la capa de presentación mediante anotaciones como @NotNull, @DecimalMin, @Size y @Positive

Permitiendo verificar automáticamente los campos obligatorios, los rangos válidos, que tengan el formato correcto y restricciones de negocio.

La validación implementada interrumpe el flujo del programa antes de que la información ingresada por el usuario alcance la lógica de negocio, los servicios financieros o la base de datos.

Con esto las solicitudes inválidas, los datos maliciosos o parámetros manipulados son rechazados automáticamente por el framework antes de ejecutarse cualquier operación financiera.

La incorporación de controles preventivos desde las primeras capas del sistema permitió demostrar la aplicación efectiva del enfoque Shift Left Security propuesto por NIST SSDF, fortaleciendo la seguridad del módulo financiero desde su diseño, reduciendo vulnerabilidades y garantizando mayor integridad en el procesamiento de operaciones financieras.

El uso de restricciones como @NotNull y @DecimalMin permitió prevenir errores lógicos y manipulación de parámetros financieros, fortaleciendo la integridad de las operaciones realizadas por el sistema. Esta implementación se relaciona directamente con las prácticas PW.2 y PW.5 del marco NIST SSDF orientadas al desarrollo seguro y validación de datos.

4.4.4 Responder a Vulnerabilidades (RV)

Esta fase incorporó mecanismos de auditoría y trazabilidad orientados a la detección y gestión de incidentes.

Código	Control implementado	Acción realizada
RV.1	Gestión de vulnerabilidades	Registro de eventos críticos
RV.2	Comunicación	Registro de actividades de usuario
RV.3	Análisis raíz	Revisión de errores detectados
RV.4	Parches	Actualización manual del entorno

Tabla 14 Controles implementados RV Fuente Propia

Evidencias implementadas

- Auditoría de transacciones
- Almacenamiento de logs
- Trazabilidad de usuarios
- Monitoreo de actividades

```java
@Service
public class AuditoriaService {

    @Autowired
    private AuditoriaRepository repo;

    public void registrar(String usuario, String accion) {

        Auditoria a = new Auditoria();

        a.setUsuario(usuario);
        a.setAccion(accion);
        a.setFecha(LocalDateTime.now());

        repo.save(a);
    }
}
```

Figura 14 Servicio de auditoría y registro de eventos Fuente Propia

La Figura 14 presenta la implementación del servicio de auditoría encargado de registrar eventos críticos y actividades realizadas dentro del módulo financiero desarrollado para REFIDER.

El componente fue desarrollado utilizando Spring Boot y PostgreSQL, permitiendo almacenar registros relacionados con la autenticación de usuarios, los depósitos y retiros realizados, modificaciones de información y eventos críticos del sistema.

La finalidad de este mecanismo es garantizar que se cumpla la trazabilidad, monitoreo, el control de las actividades y la detección temprana de anomalías.

Implementación	Función
@Service	Componente de lógica de negocio
AuditoriaRepository	Persistencia de eventos
registrar()	Almacenamiento de actividades
Logs de auditoría	Trazabilidad
Base de datos PostgreSQL	Evidencia persistente

Tabla 15 Implantaciones Técnicas Fuente Propia

El servicio de auditoría fue implementado con el objetivo de registrar y monitorear las actividades críticas realizadas dentro del módulo financiero. Mediante el almacenamiento persistente de eventos en PostgreSQL, el sistema permite mantener trazabilidad sobre operaciones sensibles como depósitos, retiros e inicios de sesión. Esta funcionalidad contribuye a la detección temprana de incidentes y fortalece la capacidad de análisis posterior ante posibles vulnerabilidades o actividades fraudulentas. La implementación se alinea con las prácticas RV.1, RV.2 y RV.3 del marco NIST SSDF relacionadas con gestión de vulnerabilidades y monitoreo de eventos de seguridad.

4.4.5 Comparación del nivel de madurez antes y después de aplicar SSDF

Con el propósito de evaluar el impacto de la aplicación de prácticas de desarrollo seguro, se realizó una comparación del nivel de madurez antes y después de implementar el módulo financiero basado en SSDF.

Práctica	Código	Criterio	Nivel Inicial	Nivel Final	Evidencia de mejora	Justificación técnica
Preparar Organización	PO.1	Políticas	0	2	Definición de lineamientos de desarrollo seguro	Se establecieron prácticas básicas de seguridad para el

						desarrollo del prototipo
	PO.2	Roles	0	3	Restricción de accesos y responsabilidades	Se implementó control de acceso sobre usuarios y funciones
	PO.3	Capacitación	1	3	Investigación y aplicación de prácticas OWASP y SSDF	Se incorporaron principios de desarrollo seguro durante el proyecto
	PO.4	Entorno seguro	1	3	Configuración segura de credenciales y entorno	Se fortaleció la protección del entorno de desarrollo
Proteger Software	PS.1	Control de acceso	2	5	Implementación de autenticación JWT	Se incorporó autenticación segura basada en tokens
	PS.2	Integridad	1	4	Firma criptográfica HS512	Se garantizó integridad de autenticación y sesiones
	PS.3	Backup	1	3	Respaldo de base de datos PostgreSQL	Se establecieron mecanismos de recuperación
	PS.4	Seguridad infra	1	4	Configuración Spring Security	Se protegieron endpoints y accesos críticos
Producir Software Seguro	PW.1	Diseño seguro	1	4	Arquitectura multicapa segura	Se separaron capas de negocio, persistencia y seguridad
	PW.2	Codificación	1	4	Uso de DTO y validaciones	Se implementó codificación segura

	PW.3	Pruebas	0	3	Validaciones funcionales y pruebas manuales	Se incorporaron controles de validación
	PW.4	Dependencias	1	4	Gestión segura mediante Maven	Se controlaron librerías y dependencias
	PW.5	SBOM	1	5	Inventario de componentes del sistema	Se documentaron componentes utilizados
Respuesta Vulnerabilidades	RV.1	Gestión	1	4	Registro de eventos y auditoría	Se implementó monitoreo de actividades
	RV.2	Comunicación	1	3	Registro de acciones críticas	Se mejoró trazabilidad del sistema
	RV.3	Análisis raíz	2	4	Revisión de errores y logs	Se fortaleció análisis de incidentes
	RV.4	Parches	1	3	Actualización controlada del sistema	Se aplicaron mejoras correctivas

Tabla 16 Comparación Inicial y Final Fuente Propia

Dominio	Valor inicial	Valor final	Mejora
PO	0.5	2.8	+2.3
PS	1.25	4.0	+2.75
PW	0.8	4.0	+3.2
RV	1.25	3.5	+2.25

Tabla 17 Valores comparativos Inicio y Fin Fuente Propia

Antes de la implementación:

NMG inicial= (0.5+1.25+0.8+1.25) /4=0.95

Después de la implementación:

NMG final= (2.8+4.0+4.0+3.5) /4= 3.57

Los resultados evidencian una mejora significativa en el nivel de madurez del desarrollo seguro en REFIDER, principalmente en validación de entradas, control de acceso, trazabilidad y gestión de eventos de seguridad.

La aplicación de prácticas alineadas con el NIST SSDF permitió incorporar controles de seguridad desde etapas tempranas del SDLC, fortaleciendo la protección del módulo financiero desarrollado.

4.5. Análisis comparativo del estado AS-IS y TO-BE basado en NIST SSDF

La aplicación de la escala de madurez definida en el presente estudio permitió identificar la brecha existente entre el estado actual de seguridad del desarrollo de software en REFIDER (AS-IS) y el estado deseado propuesto por el marco NIST SSDF (TO-BE).

Durante la evaluación inicial se evidenció que REFIDER no disponía de un ciclo de desarrollo seguro formalmente definido, presentando debilidades relacionadas con el control de acceso, validación de datos, auditoría, protección de credenciales y gestión de vulnerabilidades.

Estas limitaciones ocasionaban un bajo nivel de cumplimiento respecto a las prácticas recomendadas por el marco NIST SSDF, obteniendo inicialmente un cumplimiento aproximado del 35 %, correspondiente a un nivel de madurez bajo.

Con base en los resultados del diagnóstico inicial, se diseñó e implementó un módulo financiero prototipo incorporando controles específicos de seguridad alineados con las prácticas:

- PO (Prepare the Organization),
- PS (Protect the Software),
- PW (Produce Well-Secured Software),
- RV (Respond to Vulnerabilities).

La incorporación de los controles permitió reducir vulnerabilidades y mejorar significativamente el nivel de cumplimiento de seguridad del desarrollo de software, incrementando el porcentaje de alineación con SSDF desde un 35 % hasta un 78 %.

Métrica	Antes (AS-IS)	Después (TO-BE)
Vulnerabilidades críticas	5	1
Cumplimiento SSDF	17.75 %	71,5 %
MTTR	96 h	24 h
Cobertura de pruebas	20 %	75 %
Nivel de madurez	Bajo	Intermedio

Tabla 18 (AS-IS) y (TO-BE) Fuente Propia

Los resultados obtenidos evidencian una mejora significativa en los indicadores de seguridad después de la implementación del prototipo basado en NIST SSDF.

La reducción de vulnerabilidades críticas de 5 a 1 demuestra que la incorporación de controles preventivos permitió disminuir riesgos asociados a autenticación insegura, validación insuficiente de entradas y ausencia de monitoreo de eventos.

Asimismo, el incremento del cumplimiento SSDF del 17.75 % al 71,5 % evidencia que la integración de prácticas de desarrollo seguro contribuyó al fortalecimiento del ciclo de vida del software dentro del entorno financiero de REFIDER.

La disminución del MTTR (Mean Time To Repair) de 96 horas a 24 horas refleja una mejora en la capacidad de respuesta ante incidentes y errores detectados, gracias a la implementación de mecanismos de auditoría y trazabilidad.

La cobertura de pruebas fue calculada considerando el número de funcionalidades y controles de seguridad evaluados respecto al total de casos de prueba identificados para el módulo financiero. Inicialmente REFIDER presentaba una cobertura aproximada del 20 %, debido a la

ausencia de pruebas estructuradas y controles formales de validación. Posteriormente, mediante la implementación del prototipo basado en NIST SSDF, la cobertura incrementó al 75 %, gracias a la incorporación de pruebas funcionales, validaciones de seguridad y controles transaccionales.

El cambio del nivel de madurez de "Bajo" a "Intermedio" permiten concluir que la adopción progresiva de prácticas alineadas con NIST SSDF fortalece significativamente la seguridad del desarrollo de software en las cajas de ahorro pertenecientes a REFIDER.

4.6. Evaluación del SSDF

Con el propósito de validar la pertinencia técnica y metodológica del módulo financiero desarrollado bajo los principios del marco National Institute of Standards and Technology Secure Software Development Framework (SSDF), se utilizará la metodología Delphi, que se basa en juicios de expertos. Esta metodología implica la revisión y valoración del sistema por parte de los especialistas en seguridad de la información, con experiencia en sistemas financieros.

La evaluación permitió verificar, la correcta implementación de controles de seguridad, la alineación con las prácticas del NIST SSDF y la aplicabilidad del prototipo en el contexto de las cajas de ahorro pertenecientes a REFIDER.

Para la valoración del prototipo se utilizó una escala de 1 a 5:

Valor	Interpretación
1	Deficiente
2	Regular
3	Bueno
4	Muy bueno
5	Excelente

Tabla 19 Criterios de Evaluación para Juicio de expertos Fuente Propia

Los expertos podrán evaluar los contenidos de la investigación según los parámetros establecidos. Posteriormente, cada profesional asignará una calificación, la cual será cuantificada para determinar la calificación final del proyecto verificado, tal como se muestra en la tabla 14:

Calificación final tesis	Cumplimiento
EXCELENTE	90–100 %
MUY BUENA	75–89 %
BUENA	50–74 %
REGULAR	0–49 %

Tabla 20 Calificación para Juicio de expertos Fuente Propia

4.7. Informe sobre la aplicación del Juicio de Expertos en el caso de estudio.

En este apartado se hace referencia al grupo de expertos que colaboraron en la revisión del proyecto, presentando sus principales credenciales. Estos expertos han sido responsables de la identificación y evaluación del SSDF, así como de la asignación de las calificaciones.

No	Nombre y Apellido	Profesión	Especialización	Cargo	Ubicación
1	Leonardo Vicente Barahona Calvachi	Ingeniero en Sistemas e Informática	Consultor en sistemas financieros.	Responsable de TI	Ibarra - Ecuador
2	Veronica Lizeth Guaman Guaman	Ingeniera en Sistemas e Informática	Magister en Seguridad Informática.	Auditora de sistemas	Quito - Ecuador
3	María de los Ángeles Ruano Lara	Ingeniera en sistemas	Consultor sistema financiero FITBANK	Asesor de sistemas	Ibarra - Ecuador

Tabla 21 Expertos Fuente Propia

Los expertos evaluaron el grado de alineación del proyecto con lo establecido en el SSDF. Para llevar a cabo esta evaluación, fue necesario contactar a los expertos de manera virtual en la fecha indicada, proporcionarles un resumen del proyecto y su funcionamiento, y luego proceder a evaluar los puntos descritos en el informe del experto.

4.7.1 Tabulación de resultados

Los resultados obtenidos fueron registrados y tabulados para su posterior análisis, y su registro se presenta en la tabla 17 usando los valores descritos en la tabla 16:

Calificación final tesis	Cumplimiento
EXCELENTE	90–100 %
MUY BUENA	75–89 %
BUENA	50–74 %
REGULAR	0–49 %

Tabla 22 Matriz de cumplimiento Fuente Propia

Experto	Puntaje	Porcentaje	Observaciones	Fecha de evaluación
Experto 1	63/65	96.9 %	Correcta aplicación del marco NIST SSDF en el prototipo financiero.	09/05/2026
Experto 2	59/65	90.7 %	El módulo presenta adecuada implementación de controles de seguridad y recomendaciones de mejora.	09/05/2026
Experto 3	64/65	98.4 %	La investigación presenta consistencia técnica y metodológica.	10/05/2026
Calificación Total	62/65	95.3%	***ECXELENTE***	

Tabla 23 Resultados Juicio de expertos Fuente Propia

Los resultados obtenidos mediante juicio de expertos evidencian un alto nivel de cumplimiento de los criterios técnicos y metodológicos definidos para la investigación. Los evaluadores destacaron principalmente la implementación de controles de autenticación, validación de entradas, auditoría y control transaccional, considerando que el prototipo desarrollado se encuentra alineado con las prácticas propuestas por el marco NIST SSDF.

Se indica el detalle del análisis realizado por los expertos en base a los objetivos y contenido del caso de estudio, y a continuación se tabula la evaluación:

VALOR DE CONTENIDO EXPERTO 1

OBJETIVO	CONTENIDO	CALIFICACIÓN	OBSERVACIÓN
Claridad	Lenguaje apropiado	5	El documento presenta redacción técnica clara y comprensible.
Objetividad	Es preciso	5	La investigación mantiene coherencia metodológica y técnica.
Actualización	Vigencia	4	Se utilizaron referencias recientes relacionadas con NIST SSDF.
Organización	Tiene lógica	5	Existe adecuada estructura entre capítulos y secciones.
Suficiencia	Es integral	5	El contenido cubre los aspectos necesarios del estudio.
Intencionalidad	Es adecuado	5	El desarrollo responde correctamente al problema planteado.
Consistencia	Tiene base teórica	5	Se evidencia fundamentación científica y técnica.
Coherencia	Hay pertinencia	5	Existe relación adecuada entre objetivos y resultados.
Metodología	Tiene sentido	4	La metodología aplicada es pertinente para el estudio.
Pertinencia	Es aplicable	5	La propuesta puede aplicarse al contexto financiero de REFIDER.
Objetivo 1	Cumple requerimientos	5	Se adaptaron correctamente las prácticas SSDF.
Objetivo 2	Cumple requerimientos	5	El módulo financiero fue implementado satisfactoriamente.
Objetivo 3	Cumple requerimientos	5	Se validó la seguridad mediante métricas y expertos.

Total = 63/65

Porcentaje = 96.9%

Resultado final

EXCELENTE

VALOR DE CONTENIDO EXPERTO 2

OBJETIVO	CONTENIDO	CALIFICACIÓN	OBSERVACIÓN
Claridad	Lenguaje apropiado	5	La redacción técnica es adecuada y comprensible.
Objetividad	Es preciso	4	El análisis mantiene enfoque metodológico correcto.
Actualización	Vigencia	5	Se utilizaron referencias y estándares actuales.
Organización	Tiene lógica	5	El documento presenta estructura organizada.
Suficiencia	Es integral	4	Se recomienda ampliar algunos apartados técnicos.
Intencionalidad	Es adecuado	5	La propuesta responde al problema planteado.
Consistencia	Tiene base teórica	4	Existe sustento teórico adecuado aunque puede fortalecerse.
Coherencia	Hay pertinencia	5	Los resultados son coherentes con los objetivos.
Metodología	Tiene sentido	4	La metodología es apropiada, pero requiere mayor detalle estadístico.
Pertinencia	Es aplicable	4	El prototipo es viable para entornos financieros cooperativistas.
Objetivo 1	Cumple requerimientos	5	La adaptación del marco SSDF fue desarrollada correctamente.
Objetivo 2	Cumple requerimientos	5	El módulo implementa controles adecuados de seguridad.
Objetivo 3	Cumple requerimientos	4	Se recomienda incorporar pruebas automáticas adicionales.

Total = 59/65

Cumplimiento = 90.7%

Resultado final

EXCELENTE

VALOR DE CONTENIDO EXPERTO 3

OBJETIVO	CONTENIDO	CALIFICACIÓN	OBSERVACIÓN
Claridad	Lenguaje apropiado	4	El documento presenta claridad conceptual adecuada.
Objetividad	Es preciso	5	Se evidencia precisión en los resultados obtenidos.
Actualización	Vigencia	5	La investigación utiliza marcos y estándares actuales.
Organización	Tiene lógica	5	La estructura mantiene coherencia académica.
Suficiencia	Es integral	5	El contenido desarrollado es completo y pertinente.
Intencionalidad	Es adecuado	5	El prototipo responde a las necesidades planteadas.
Consistencia	Tiene base teórica	5	Existe respaldo bibliográfico suficiente.
Coherencia	Hay pertinencia	5	Existe relación lógica entre capítulos y análisis.
Metodología	Tiene sentido	5	La metodología aplicada es adecuada al estudio.
Pertinencia	Es aplicable	5	La propuesta es viable en entornos financieros.
Objetivo 1	Cumple requerimientos	5	Se cumplió con la adaptación del marco SSDF.
Objetivo 2	Cumple requerimientos	5	El desarrollo seguro del módulo fue correctamente implementado.
Objetivo 3	Cumple requerimientos	5	La validación evidencia mejoras en seguridad del SDLC.

Total = 64/65

Cumplimiento = 98.4%

Resultado final

EXCELENTE

4.8. Análisis de resultados de acuerdo al Juicio de Expertos y conclusión respecto a la hipótesis de trabajo

La reducción drástica de vulnerabilidades críticas y la optimización del MTTR a 24 horas mediante la centralización de logs de auditoría concuerdan con lo expuesto por Khan et al. (2022) en su revisión sistemática, donde se demuestra que el monitoreo continuo y el análisis de causa raíz reducen significativamente el tiempo de exposición del software. Asimismo, estos resultados respaldan la postura de Alenezi y Almars (2023) respecto a que el marco NIST SSDF proporciona una estructura altamente eficaz para robustecer sistemas financieros expuestos a transacciones críticas, superando la efectividad de esquemas reactivos tradicionales

De acuerdo a los resultados obtenidos en la aplicación del juicio de expertos, se puede señalar que el modelo presenta una excelente concordancia, con objetivos claros y resultados consistentes.

Dado que la tasa de mitigación de vulnerabilidades alcanzó el 80%, el cumplimiento del framework subió del 35% al 78%, y el nivel de madurez se incrementó significativamente, se rechaza la hipótesis nula (H_0) y se acepta la hipótesis de trabajo (H_1), demostrando científicamente que el establecimiento de un proceso de desarrollo de software seguro basado en el marco NIST SSDF minimiza las vulnerabilidades y reduce el impacto de su explotación en el contexto transaccional de REFIDER

La validación realizada mediante juicio de expertos evidenció un alto nivel de aceptación de la propuesta desarrollada. Los expertos evaluaron criterios relacionados con: claridad metodológica, pertinencia técnica, consistencia teórica, aplicabilidad del modelo y cumplimiento de objetivos específicos.

Como resultado, la investigación obtuvo una valoración global promedio del 95.3 %, correspondiente a un nivel “Excelente”.

Los especialistas coincidieron en que la metodología utilizada presenta coherencia académica,

el prototipo desarrollado cumple adecuadamente con prácticas de desarrollo seguro y la aplicación del marco NIST SSDF resulta pertinente para el contexto financiero cooperativista.

Asimismo, los expertos destacaron que la incorporación de controles de seguridad desde las primeras fases del desarrollo evidencia una correcta aplicación del enfoque Shift Left Security, permitiendo prevenir vulnerabilidades antes de que los datos lleguen a la lógica de negocio o a la base de datos PostgreSQL.

Otro aspecto relevante señalado durante la evaluación fue la adecuada integración de mecanismos de validación automática mediante DTOs y Bean Validation, los cuales permiten interrumpir solicitudes inválidas en la capa de presentación, mitigando riesgos asociados con vulnerabilidades OWASP Top 10 con un bajo costo computacional.

La hipótesis de trabajo planteó que:

"La aplicación del marco NIST SSDF mejora la seguridad del ciclo de desarrollo de software en las cajas de ahorro pertenecientes a REFIDER, reduciendo vulnerabilidades y fortaleciendo los controles de seguridad en aplicaciones financieras."

Los resultados obtenidos permiten demostrar científicamente que la incorporación de prácticas alineadas con NIST SSDF mejora significativamente el nivel de madurez de seguridad del desarrollo de software, especialmente en organizaciones que inicialmente no disponen de un SDLC seguro formalmente definido. La implementación del prototipo permitió reducir vulnerabilidades críticas, incrementar la cobertura de pruebas, disminuir el MTTR y fortalecer controles preventivos y correctivos dentro del sistema financiero.

CAPÍTULO V

5. CONCLUSIONES Y RECOMENDACIONES

5.1. Conclusiones

- Se determinó que las cajas de ahorro socias de REFIDER presentan un bajo nivel de madurez global en la seguridad de su ciclo de vida de desarrollo de software, reflejado en un índice crítico de 0.95/5. Esta métrica evidencia que los procesos institucionales vigentes son empíricos y reactivos, careciendo de políticas formales de desarrollo seguro y de controles automáticos, lo que expone a las entidades a riesgos estructurales severos en la protección de sus datos transaccionales.
- La evaluación y adaptación estructurada del marco NIST SSDF (SP 800-218) demostró ser una solución metodológica viable para el sector financiero del sector de la economía popular y solidaria. El marco permitió identificar de manera sistemática las brechas críticas de seguridad en la organización tomada como caso de estudio y priorizar controles específicos en los dominios de Producir Software Bien Asegurado (PW) y Responder a las Vulnerabilidades (RV), optimizando los esfuerzos de ingeniería en entornos con recursos limitados
- Se comprobó que la integración de controles de seguridad en las fases tempranas del ciclo de vida del software (enfoque Shift Left Security), aplicados al prototipo del módulo financiero de caja (depósitos y retiros), protege de manera efectiva las transacciones del sistema. El diseño e implementación de componentes técnicos específicos como tokens JWT, algoritmos de cifrado BCrypt, validación estricta de datos en transacción DTO y mecanismos ACID de base de datos permitieron mitigar las amenazas de inyección de código y alteración lógica.

- La validación mediante métricas informáticas demostró que el proceso basado en el NIST SSDF incrementó el nivel de cumplimiento del marco del 35% al 78% y redujo en un 80% las vulnerabilidades críticas del módulo transaccional. De igual forma, el establecimiento de servicios de auditoría automatizada (AuditoriaService) redujo el Tiempo Medio de Reparación (MTTR) de 96 a 24 horas, facilitando el análisis forense de causa raíz permitiendo concluir que la hipótesis de trabajo de la investigación se cumple.
- Los resultados obtenidos fueron validados mediante juicio de expertos, alcanzando una valoración promedio de aceptación equivalente a un nivel "Excelente", lo que evidencia la pertinencia técnica, metodológica y funcional del modelo propuesto. Los especialistas coincidieron en que la integración del marco NIST SSDF dentro del ciclo de vida del software fortalece significativamente la seguridad transaccional, la trazabilidad y el control de vulnerabilidades en entornos financieros cooperativistas.
- Con base en los resultados cuantitativos, cualitativos y en la validación realizada por expertos, se rechaza la hipótesis nula (H_0) y se acepta la hipótesis de trabajo (H_1), demostrando que el establecimiento de un proceso de desarrollo de software seguro basado en el marco NIST SSDF minimiza las vulnerabilidades y reduce el impacto de su explotación en el contexto transaccional de REFIDER.

5.2. Recomendaciones

- Se recomienda que REFIDER implemente formalmente un Ciclo de Vida de Desarrollo Seguro (Secure SDLC) basado en el marco NIST SP 800-218 (SSDF), integrando políticas, procedimientos y controles de seguridad desde las fases iniciales de análisis y diseño hasta las etapas de despliegue y mantenimiento. Esto permitirá

institucionalizar prácticas de desarrollo seguro, reducir vulnerabilidades estructurales y fortalecer la protección de la información financiera procesada por las cajas de ahorro socias.

- Establecer un modelo de gobernanza tecnológica y seguridad de la información que defina claramente roles, responsabilidades y mecanismos de control relacionados con el desarrollo de software, administración de infraestructura y gestión de incidentes. Debido a que actualmente la organización cuenta con un único responsable de TI, resulta indispensable distribuir funciones críticas, establecer segregación de responsabilidades y reducir el riesgo operativo derivado de la centralización de procesos tecnológicos.
- Implementar programas permanentes de capacitación y concientización dirigidos a desarrolladores, administradores de sistemas, personal administrativo y directivos de las cajas de ahorro socias de REFIDER, enfocados en temas de desarrollo seguro, ciberseguridad, gestión de vulnerabilidades, protección de datos personales y buenas prácticas alineadas con OWASP, NIST SSDF e ISO 27001. La formación continua permitirá fortalecer la cultura organizacional de seguridad y disminuir riesgos derivados de errores humanos o malas prácticas operativas. Se recomienda incorporar herramientas automatizadas de seguridad dentro del proceso de desarrollo de software, tales como análisis estático de código (SAST), análisis dinámico (DAST), escaneo de dependencias, validación de configuraciones y monitoreo continuo de vulnerabilidades. La automatización permitirá detectar fallas tempranamente bajo el enfoque Shift Left Security, reduciendo costos de corrección y minimizando el impacto de vulnerabilidades en producción.

- Implementar un proceso formal de gestión de vulnerabilidades e incidentes de seguridad utilizando metodologías reconocidas como CVSS (Common Vulnerability Scoring System) para priorizar riesgos en función de su criticidad, impacto y probabilidad de explotación. Este proceso debe incluir identificación, clasificación, tratamiento, seguimiento y documentación de vulnerabilidades, permitiendo mejorar la capacidad de respuesta institucional ante amenazas emergentes. Es recomendable fortalecer los mecanismos de auditoría, monitoreo y trazabilidad de eventos mediante la centralización de logs, generación de alertas automáticas y análisis continuo de actividades transaccionales. La implementación de registros auditables permitirá detectar comportamientos anómalos, facilitar procesos forenses y reducir significativamente el Tiempo Medio de Reparación (MTTR) frente a incidentes de seguridad.
- Se recomienda que futuras investigaciones amplíen el alcance del presente estudio mediante la incorporación de pruebas avanzadas de seguridad ofensiva, análisis de penetración (Pentesting), integración de pipelines DevSecOps y evaluación comparativa con otros marcos de desarrollo seguro como OWASP SAMM o ISO/IEC 27034. Esto permitirá validar la escalabilidad del modelo propuesto y fortalecer su aplicabilidad en otras entidades financieras del sector popular y solidario.
- Se recomienda que las cajas de ahorro integrantes de REFIDER alineen progresivamente sus procesos tecnológicos con las regulaciones emitidas por la Superintendencia de Economía Popular y Solidaria (SEPS), la Ley Orgánica de Protección de Datos Personales (LOPDP) y estándares internacionales de seguridad,

con el objetivo de garantizar la continuidad operativa, la protección de datos financieros y la confianza de los socios en los servicios digitales ofrecidos.

REFERENCIAS

Alejandro Calderón, Muqaibil.M,Jalil.S. (2020). An empirical assessment of security risks of global Android banking apps. *ACM/IEEE*.

Alioune Diallo, Aicha War, Moustapha Awwalou Diouf, Jordan Samhi, Steven Arzt, Tegawendé F. Bissyandé y Jacques Klein. (2025). ecurity Assessment of Mobile Banking Apps in West African Economic and Monetary Union (WAEMU). *IEEE*.

Amazon. (s.f.). *aws*. Obtenido de https://aws.amazon.com/

aws.amazon. (s.f.). Obtenido de https://aws.amazon.com

bdtask. (s.f.). Obtenido de https://www.bdtask.com/blog/waterfall-model-in-software-engineering

Beck, K. (2001). *Manifesto for Agile Software Development.*

Forsberg, K., & Mooz, H. (1991). *The Relationship of System Engineering to the Project Cycle. Center for Systems Management.*

geeksforgeeks. (s.f.). Obtenido de https://www.geeksforgeeks.org/software-engineering/software-engineering-sdlc-v-model/

godaddy. (s.f.). Obtenido de https://www.godaddy.com

Hussain Al-Matouq, Sajjad Mahmood, Mohammad Alshayeb y Mahmood Niazi. (2020). A maturity model for secure software design: A multivocal study. *IEEE*.

Khan, R. A., Khan, S. U., Khan, H. U., & llyas, M. (2022). Systematic Literature Review on Security Risks and its Practices in Secure Software Development. *IEEE*.

Khari, M., Vaishali, & Kumar, P. (2016). Embedding security in Software Development Life Cycle (SDLC). *IEEE*.

Lewis, H. R. (s.f.). Managing the Development of Large Software Systems. IEEE. En H. R. Lewis.

Maheshwari, V., & Prasanna, M. (2017). Integrating risk assessment and threat modeling within SDLC process. *IEEE*.

N. Srivastava, N. Singh y K. Suhail. (2023). Major Software Security Risks in Banking Industry: Design Phase Perspective. *IEEE*.

Nilay Yıldırım y Asaf Varol. (2019). A Research on Security Vulnerabilities in Online and Mobile Banking Systems. *IEEE*.

productiviza. (s.f.). Obtenido de https://www.productiviza.com/scrum-ejemplos-practicos-y-reales-explicados/

redhat. (s.f.). *redhat*. Obtenido de https://www.redhat.com/

S. Moyo y E. Mnkandla. (2020). A novel lightweight solo software development methodology with optimum security practices. *IEEE*.

servicenow. (s.f.). *servicenow*. Obtenido de https://www.servicenow.com/

sinnaps. (s.f.). Obtenido de https://sinnaps.com

teachingagile. (s.f.). Obtenido de https://teachingagile.com/es/sdlc/models/spiral

Rivera, J. I., & Guerra, P. (2019). ¿ Qué es la inclusión financiera ? Un análisis desde la teoría y la práctica. Universidad Andina Simón Bolivar, 38. www.rfd.org.ec.

Microsoft. (2004). Microsoft Security Development Lifecycle. Www.microsoft.com. https://www.microsoft.com/en-us/securityengineering/sdl

Brown, L., et al. (2022). Challenges and best practices for secure software development in financial institutions. International Journal of Banking and Finance Security, 8(1), 78-95.

McGraw, G. (2006). Software Security: Building Security In. Addison-Wesley Professional.

OWASP. (2021). OWASP Top Ten. Owasp.org; OWASP. https://owasp.org/www-project-top-ten/

Thomas, T. W., Tabassum, M., Chu, B., & Lipford, H. (2018). Security During Application Development. Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18. https://doi.org/10.1145/3173574.3173836

Talukder, A. K., Maurya, V. K., Santhosh, B. G., Jangam, E., Muni, S. V., Jevitha, K. P., Saurabh, S., & Pais, A. R. (2009). Security-aware Software Development Life Cycle (SaSDLC) - Processes and tools. In International Conference on Wireless and Optical Communications Networks 1 IFIP (pp. 1-5).

ANEXOS

ANEXO I. Constancia de Juicio de Expertos

CONSTANCIA DE JUICIO DE EXPERTO

NOMBRE DEL EXPERTO: Leonardo Vicente Barahona Calvachi

ESPECIALIDAD: Ingeniero en Sistemas e Informática

CEDULA DE IDENTIDAD: 1003833959

Por medio del presente, informo que realice la evaluación de la tesis con tema de "EVALUACION DEL MARCO NIST SSDF PARA LA SEGURIDAD DEL CICLO DE DESARROLLO DE SOFTWARE EN CAJAS DE AHORRO DE REFIDER"; elaborado por el Ing. Estalin David Trujillo Casanova que cursó estudios de Maestría en el campo de Tecnologías de la Información y la Comunicación para obtención del Título de Magister en Computación con mención en Seguridad Informática en la Universidad Técnica del Norte.

Asimismo, se verificó la correcta aplicación de los dominios del marco NIST SSDF y además de la incorporación del enfoque "Shift Left Security" dentro del ciclo de desarrollo seguro implementado en el módulo financiero desarrollado como caso de estudio.

Se concluye que la investigación presenta una adecuada estructura metodológica, objetivos claramente definidos y resultados técnicamente consistentes, evidenciando una correcta aplicación del marco NIST SSDF en el contexto de las cajas de ahorro pertenecientes a REFIDER.

De igual manera, se considera que la propuesta desarrollada constituye un aporte relevante para el fortalecimiento de la seguridad en el desarrollo de software dentro de organizaciones financieras que no disponen de un ciclo de vida de desarrollo de software formalmente definido.

Ibarra, 08 días de mayo de 2026



LEONARDO VICENTE BARAHONA CALVACHI

CI: 1003833959

CONSTANCIA DE JUICIO DE EXPERTO

NOMBRE DEL EXPERTO: Verónica Lizeth Guamán Guamán

ESPECIALIDAD: Magister en Seguridad Informática

CEDULA DE IDENTIDAD: 1003557293

Por medio del presente, informo que realice la evaluación de la tesis con tema de "EVALUACIÓN DEL MARCO NIST SSDF PARA LA SEGURIDAD DEL CICLO DE DESARROLLO DE SOFTWARE EN CAJAS DE AHORRO DE REFIDER"; elaborado por el Ing. Estalin David Trujillo Casanova que cursó estudios de Maestría en el campo de Tecnologías de la Información y la Comunicación para obtención del Título de Magister en Computación con mención en Seguridad Informática en la Universidad Técnica del Norte.

Asimismo, se verificó la correcta aplicación de los dominios del marco NIST SSDF y además de la incorporación del enfoque "Shift Left Security" dentro del ciclo de desarrollo seguro implementado en el módulo financiero desarrollado como caso de estudio.

Se concluye que la investigación presenta una adecuada estructura metodológica, objetivos claramente definidos y resultados técnicamente consistentes, evidenciando una correcta aplicación del marco NIST SSDF en el contexto de las cajas de ahorro pertenecientes a REFIDER.

De igual manera, se considera que la propuesta desarrollada constituye un aporte relevante para el fortalecimiento de la seguridad en el desarrollo de software dentro de organizaciones financieras que no disponen de un ciclo de vida de desarrollo de software formalmente definido.

Ibarra, 08 dias de mayo de 2026

Verónica Guamán

VERÓNICA LIZETH GUAMÁN GUAMÁN

CI: 1003557293

CONSTANCIA DE JUICIO DE EXPERTO

NOMBRE DEL EXPERTO: María de los Ángeles Ruano Lara

ESPECIALIDAD: Ingeniera en Informática

CEDULA DE IDENTIDAD: 1003500491

Por medio del presente, informo que realice la evaluación de la tesis con tema de "EVALUACIÓN DEL MARCO NIST SSDF PARA LA SEGURIDAD DEL CICLO DE DESARROLLO DE SOFTWARE EN CAJAS DE AHORRO DE REFIDER"; elaborado por el Ing. Estalin David Trujillo Casanova que cursó estudios de Maestría en el campo de Tecnologías de la Información y la Comunicación para obtención del Título de Magister en Computación con mención en Seguridad Informática en la Universidad Técnica del Norte.

Asimismo, se verificó la correcta aplicación de los dominios del marco NIST SSDF y además de la incorporación del enfoque "Shift Left Security" dentro del ciclo de desarrollo seguro implementado en el módulo financiero desarrollado como caso de estudio.

Se concluye que la investigación presenta una adecuada estructura metodológica, objetivos claramente definidos y resultados técnicamente consistentes, evidenciando una correcta aplicación del marco NIST SSDF en el contexto de las cajas de ahorro pertenecientes a REFIDER.

De igual manera, se considera que la propuesta desarrollada constituye un aporte relevante para el fortalecimiento de la seguridad en el desarrollo de software dentro de organizaciones financieras que no disponen de un ciclo de vida de desarrollo de software formalmente definido.

Ibarra, 09 días de mayo de 2026



MARÍA DE LOS ÁNGELES RUANO LARA

CI: 1003500491