FACULTAD DE ARQUITECTURAS E INGENIERÍAS

Trabajo de fin de carrera titulado

"ANÁLISIS COMPARATIVO DE LAS VENTAJAS Y DESVENTAJAS DEL USO DE CODIFICACIÓN SEGURA PARA UNA PEQUEÑA EMPRESA DE DESARROLLO DE SOFTWARE ECUATORIANA"

Realizado Por:

Carlos Andrés Guamán Guañuna

Director de Proyecto:

Ing. Diego Fernando Riofrío Luzcando, PHD

Como requisito para la obtención del título de: MAGISTER EN TECNOLOGÍAS DE LA INFORMACIÓN CON MENCIÓN EN SEGURIDAD DE REDES Y COMUNICACIÓN

Quito, septiembre de 2019

DECLARACIÓN JURAMENTADA

Yo, **CARLOS ANDRÉS GUAMÁN GUAÑUNA** con cédula de identidad **1719349241**, declaro bajo juramento que el trabajo aquí desarrollado es de mi autoría, que no ha sido previamente presentado para ningún grado a calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración, cedo los derechos de propiedad intelectual correspondientes a este trabajo, a la UNIVERSIDAD INTERNACIONAL SEK, según lo establecido por la Ley de Propiedad Intelectual, por su reglamento y por la normativa institucional vigente.

CARLOS ANDRÉS GUAMÁN GUAÑUNA

CI: 1719339241

DECLARATORIA

El presente trabajo de investigación titulado:

ANÁLISIS COMPARATIVO SOBRE LAS VENTAJAS Y DESVENTAJAS DEL USO DE CODIFICACIÓN SEGURA PARA UNA PEQUEÑA EMPRESA DE SOFTWARE ECUATORIANA

Realizado Por:

CARLOS ANDRÉS GUAMÁN GUAÑUNA

Como requisito para la obtención del título de:

MAGISTER EN TECNOLOGÍAS DE LA INFORMACIÓN CON MENCIÓN EN SEGURIDAD DE REDES Y COMUNICACIÓN

Ha sido dirigido por el profesor:

ING. DIEGO FERNANDO RIOFRIO LUZCANDO, PhD

Quien considera que constituye un trabajo original de su autor

Ing. Diego Fernando Riofrío Luzcando, PhD

DIRECTOR

PROFESORES INFORMANTES

Después de revisar el trabajo presentado, lo ha calificado como apto para su defensa oral ante el tribunal examinador.

Ing. Christian David Pazmiño Flores

Ing. Luis Fabian Hurtado Vargas

Quito, septiembre de 2019

AGRADECIMIENTO

A DIOS por cuidarme en cada instante de mi vida, por permitirme culminar otra etapa de educación formal, por darme fuerza y valor para enfrentar todas las vicisitudes que se han presentado durante este proceso de educación.

A mi familia y en especial a mis padres Andrecito y Michita por ayudarme siempre, sus consejos y muestras de aliento día a día me ayudaron a enfrentar este camino

A mi familia por siempre estar presentes, por su motivación, consejos y apoyo incondicional en el transcurso mi vida, los amo infinitamente.

A la Universidad Internacional SEK por permitirme ser parte de tan noble institución en el programa de maestría, gracias a todos los docentes por ofrecer su conocimiento y compartir su experiencia laboral durante cada clase.

A la empresa Pentaedro Digital Zone, por consentir la realización del presente estudio en sus proyectos y ofrecer la información necesaria de forma oportuna

A mi Director de Tesis, Diego Riofrío, quien, gracias a su aporte de conocimientos, me ha guiado en la realización de un trabajo bien elaborado.

A mis amigos de Maestría Evelyn, Alexis y Juan Carlos, con los cuales conformamos un buen equipo de trabajo y de su aporte llevo muchas experiencias y recuerdos

.

DEDICATORIA

A mi madre Mercedes

Por ayudarme con el primer paso para el ingreso a la Maestría, por enseñarme a valorar cada acción que se presenta en mi vida, por recordarme siempre que para conseguir un objetivo se debe trabajar duro y por brindarme su amor incondicional siempre

A mi padre Andrés

Por ofrecer su apoyo incondicional, por creer siempre en mis sueños e ilusiones, por enseñarme que en la vida hay que superarse cada día y sobre todo por la paciencia de estar a su lado todavía

A mis hermanos Nancy, Juan Pablo, William

Por ser siempre la fuente de alegrías, enseñanzas, y superación, por entregar su apoyo durante toda mi vida sin importar que harían lo que fuera para socorrerme cuando he caído

A mi novia Barbarita

Por sacrificar mucho de tu tiempo para apoyarme en momentos difíciles, por brindarme tu amor, por ayudarme a creer que se puede hacer lo que se propone, aunque cueste mucho trabajo

.

RESUMEN

La presente investigación trata de la identificación de ventajas y desventajas que puede tener el uso de codificación segura, a partir de normativas o estándares que avalen aseguramiento en el desarrollo de proyectos de software.

Esta fue realizada en una PYME del Ecuador que se dedica al desarrollo de aplicaciones de software a medida, en la cual se implementó un proyecto con codificación segura, utilizando la normativa ISO 27001 con soporte en la metodología SDL haciendo énfasis en el ciclo de vida del desarrollo, donde se tiene en cuenta el entrenamiento de la organización en aspectos de seguridad, también se estudia los requisitos y se diseña una solución que sea factible y adaptable, con aceptación por parte de los *stakeholders.*

Para establecer ventajas y desventajas se realizó una comparativa de este proyecto con otro desarrollado anteriormente sin tomar en cuenta ningún estándar de codificación segura. Para esto se tomó en cuenta métricas basadas en la función, las cuales se emplean para evaluar el esfuerzo global de métodos utilizados en cada uno de los proyectos.

Como resultado se ha obtenido el tiempo de implementación del proyecto con codificación segura fue más largo, sin embargo, en la fase de mantenimiento se reduce el número de horas invertidas para corrección de errores y en la fase de respuesta a incidentes tiene mejor control a esos eventos.

Palabras clave: Codificación segura, ISO 27001, SDL, Ciclo de vida del software, *stakeholders*, PYME

ABSTRACT

The purpose of the following research is to identify the possible advantages and disadvantages in the usage of secure code, based on established regulations or standards that endorse the security in the development of software projects.

The investigation was conducted with the help of a PYME in Ecuador, whose work focuses on the development of custom software applications. One project in particular, included the implementation of secure code using the ISO 27001 legislation and the SDL methodology to focus on the life cycle of the development of software. Furthermore, the project took into account the company's training on security aspects; as well as the study of requirements and the design of feasible and adaptable solutions approved by the stakeholders.

In order to determine the advantages and disadvantages of the secure code, a comparison was made between the projected already mentioned and a previous project developed without taking into consideration any standard security code. For this purpose, function metrics were used which are usually implemented to assess the global efforts of software development methods applied on each project developed at companies.

As a conclusion, the application of secure code in the implementation process takes longer to execute. However, it reduces the hours invested in maintenance and errors correction as it enables fast responses to incidents and provides better control of errors.

Keywords: Secure coding, ISO 27001, SDL, Software life cycle, Stakeholders, PYME

INDICE GENERAL DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

CAPÍTULO I

INTRODUCCIÓN

1.1. Planteamiento del Problema

El desarrollo de software mantiene la tendencia a actualizarse y mejorar sus buenas prácticas mediante el uso de herramientas que validen la integridad del código escrito en sus aplicaciones, por lo cual es necesario estandarizar las prácticas de seguridad empleadas en dicho desarrollo dentro de un ambiente de PYME, lo cual permitirá que se pueda emplear de manera eficiente los recursos utilizados durante el proceso de análisis, codificación e implementación del software a medida. (Crawford, Pollack, & England, 2006)

En el desarrollo de proyectos de software se puede utilizar codificación segura, la cual permite que todo el proceso sea seguro y se desempeñe de forma continua y correcta, en todo el ciclo de vida del producto de software (Charette, 2005).

Utilizando los estándares o normas como buenas prácticas de desarrollo de software se obtiene un producto de calidad, es así que la normativa ISO 27001 (2013) ayuda a considerar como parte esencial a la seguridad en todo el proceso de desarrollo y soporte incluyendo la seguridad del recurso humano, recursos tangibles y no tangibles manteniendo controles en todos sus aspectos que son aplicados desde el inicio hasta el fin del proyecto. Haller (2014), menciona que el uso de la ISO 27001:2013 es una decisión de la alta dirección, sin embargo, los desarrolladores deben invertir su tiempo para el estudio y aplicación de la misma para el mejoramiento de sus habilidades de implementación, ya que los procedimientos operativos escritos mantienen un control operativo y seguro durante todo el ciclo de vida del desarrollo del proyecto.

La normativa ISO 27002:2013 hace hincapié en los controles previos en el desarrollo y mantenimiento de aplicativos de software, es decir que existen controles que ayudan a la estabilidad en seguridad en sistemas de producción, manteniendo procedimientos de control

los cuales ayudan al mejoramiento de la calidad del producto entregado y como resultado existe una mínima posibilidad de posibles errores a corregir a futuro (Keramati, 2008)

1.1.1. Formulación del problema

La falta de una comparativa sobre las ventajas y desventajas del uso de codificación segura incide en la elección adecuada de la metodología de seguridad a usar en una pequeña empresa de software ecuatoriana.

1.2. Objetivo general

Identificar las ventajas y desventajas del uso de codificación segura mediante la revisión de la norma ISO 27001:2013 con soporte en metodología en desarrollo seguro SDL, para el sustento en la toma de decisión para la ejecución de futuros proyectos en una pequeña empresa de desarrollo de software a medida.

1.2.1 Objetivos específicos

- Estudiar los objetivos de control en el proceso de desarrollo de software con la ayuda de la ISO 27002:2013 para la inclusión de controles de seguridad y validación de datos en el desarrollo del producto.

- Implementar un sistema con codificación segura mediante el uso de los controles de la normativa 27001 (2013) con soporte en la metodología SDL para la obtención de una comparativa con un sistema sin codificación segura.

- Generar rúbricas de control de funcionamiento con la ayuda de las métricas del modelo de análisis, para la obtención de parámetros comparables entre dos proyectos.

- Comparar el sistema implementado usando la codificación segura con un sistema que no ha sido implementado con codificación segura usando la rúbrica de controles generada para la obtención de resultados la cual ayudará en la toma de decisiones en la creación de nuevos productos.

1.3. Justificación

Con la revisión de la literatura relacionada a los objetivos de la investigación planteada no se ha encontrado un estudio comparativo que permita establecer criterios de valor con respecto a ventajas y desventajas sobre la toma de decisión para la ejecución de proyectos de software.

En estudios revisados con respecto a la implementación de software se ha encontrado casos de desarrollo de software con ayuda de codificación segura y otros que no la utilizan, para este estudio la empresa provee un ejemplo sin la utilización de codificación segura por lo cual se debe implementar un factor de comparación en la cual se utilice aspectos de codificación segura, por lo que se utiliza la normativa ISO 27002:2013 con soporte en la metodología SDL, las cuales brindan herramientas de control, estandarización y calidad dentro de todo el ciclo de vida.

Con los parámetros de medición generados se elaboran documentos con valores de métricas verificables y practicas entre los dos casos. Con los datos obtenidos de las métricas se presentan cuadros comparativos en los cuales sea visible cual es el comportamiento de los productos antes, durante y después, de utilizar codificación segura, y de esta manera, proporcionar un soporte cuantitativo para la toma de decisiones de futuros proyectos.

CAPÍTULO II

MARCO TEÓRICO

2.1. Seguridad informática

Según García & Alegre (2011), los constantes cambios a los que está sometido el mundo moderno, la globalización de la economía y el gran flujo de información presente en el diario vivir de la sociedad ha coartado la necesidad de soportar estas actividades mediante el uso de herramientas que estuvieran a la par con el rápido desarrollo del ecosistema en que se vive, por tanto es así como las Tecnologías de la Información y las Comunicaciones (TIC) surgen como la gran alternativa para suplir estos requerimientos de información y desarrollo. Las TIC han facilitado de una u otra manera la forma en que las personas realizan sus actividades y se adaptan a los constantes cambios, ya que poseen el potencial para cambiar drásticamente a las personas, las organizaciones y las prácticas de negocio.

2.1.1. Principios de seguridad informática

Los pilares de la seguridad informática se basan en la forma en que debería ser preservada y administrada la información, como lo menciona Dussan (2006) en su artículo "Políticas de Seguridad Informática" el primer pilar es la integridad, que busca conservar los datos tal cual como fueron concebidos desde su origen hasta su destino, es decir, que estos no hayan sufrido modificaciones o daños en cualquier instante de su custodia; el segundo pilar es la confidencialidad, que define que la información solo puede ser accedida y utilizada por el personal autorizado para tal fin, para esto existen diferentes herramientas como el encriptamiento, protección por contraseñas, etc.

2.1.2. Clasificación de seguridad informática

La seguridad Informática en si es un conjunto de controles, prácticas y herramientas en caminadas a proteger los activos de información, estas por su naturaleza y el área como tal en la que se desempeñan se pueden agrupar en protección física y lógica.

García & Alegre. (2011), mencionan como definición de la seguridad lógica como la protección que se le da a los datos en cualquier software o sistema de información y como se accede a estos, buscando la preservación de los tres pilares de la seguridad informática mencionados con anterioridad.

Según Martinez-garcia (2019), algunos parámetros con los que debería contar cualquier sistema para garantizar la protección básica de sus datos son: un módulo de autenticación en donde solo el personal registrado y con autorización pueda acceder, de tal forma que el usuario autenticado pueda ver únicamente la información para la cual tiene permiso; debe tener niveles de control para la realización de transacciones de datos que garanticen la integridad y recepción de los mismos a través de diferentes métodos como el de encriptamiento, uso de contraseñas, etc; limitación y control de servicios y recursos, es decir que no todos tiene las mismas facultades para manejar la información, habrá quienes solo podrán consultar, otros que podrán modificarla u otros que supervisarán su manipulación.

2.2. Buenas prácticas en seguridad informáticas

Los autores de COBIT5 (2012), mencionan que las buenas prácticas en seguridad de la información es un modelo que permiten evaluar y controlar todos los aspectos de TI de una organización que van desde los sistemas de información hasta el personal que los administra, enfocado siempre a los objetivos del negocio

Su finalidad es ofrecer a la organización una herramienta que automatice y facilite el cumplimiento de los objetivos del negocio haciendo un uso óptimo de los recursos TI.

Como lo mencionan los autores Pesado, Bertone u Esponda (2013)ITIL es un conjunto de buenas prácticas para la administración de recursos y servicios de TI, con el

objetivo de lograr calidad y eficiencia en los procesos productivos de la organización a través de la mejora continua. ITIL estructura el ciclo de vida de los servicios en cinco etapas:

- La estrategia del servicio se base en la identificación de los activos estratégicos de la empresa, es decir que generan valor o que la diferencia en el mercado.
- El diseño del servicio busca explorar el portafolio de servicios identificando debilidades y fortalezas para así definir estándares y políticas que permitan mejorar la calidad de los mismos.
- La transición del servicio cubre el proceso de cambio entre la mejora de servicios existentes o la creación de nuevos servicios de ser necesario, identificados en el diseño.
- La operación del servicio busca gestionar y controlar los procesos necesarios para la puesta en producción de un servicio.
- La mejora continua del servicio busca optimizar la calidad del servicio ofrecido a través de la retroalimentación constante de la estrategia, diseño, transición y operación que faciliten la adaptación al entorno.

2.3. Sistemas Críticos

Los sistemas críticos ayudan en la verificación de la codificación segura dentro de un proceso de desarrollo de software, debido a que integra aspectos de seguridad desde las primeras fases del mismo, se conocen diferentes alternativas de integración con la ingeniería de seguridad, sin embargo, su uso depende de la norma o metodología requerida.

2.4. Proceso de software seguro

Los sistemas críticos se han centrado en definir un proceso de desarrollo de software seguro, que integra prácticas de ingeniería de la seguridad en el propio ciclo de desarrollo de software. Este proceso se basa en la captura de los requisitos de seguridad desde el propio modelo de procesos de negocio y su posterior refinamiento a lo largo de todo el proceso de

desarrollo (Maña, Ray, Sánchez, & Yagüe, 2004). Existen modelos que permiten abarcar de forma general con el proceso de software seguro, los cuales se detallan a continuación:

- Descripción semántica de los requisitos de seguridad
- Extensión de UML con requisitos de seguridad
- Desarrollo de una librería de patrones de seguridad
- Integración de los componentes anteriores dentro de una herramienta software para ofrecer un proceso de ingeniería enfocado a la seguridad.

2.5. Estándares de seguridad

2.5.1. COBIT

Es un modelo para auditar la gestión y control de los sistemas de información y tecnología orientado a todas las partes interesadas de una organización, por ejemplo, administradores, usuarios y auditores, los cuales están involucrados en el proceso del negocio. COBIT siendo un marco de referencia clasifica a los procesos de las unidades de tecnología de información en cuanto dominios los cuales se detallan a continuación.

Planificación y organización, según Von Solms (2005) esta fase cubre estrategia y tácticas referentes a la identificación en como la tecnología contribuye de mejor manera al logro de los objetivos del negocio.

Adquisición e implantación, de acuerdo a Von Solms (2005), esta fase hace referencia a los servicios requeridos y abarca operaciones tales como el entrenamiento basados en seguridad y aspectos de continuidad del negocio.

Monitoreo, según Von Solms (2005), afirma que todos los procesos deben ser evaluados regularmente para entender si los requerimientos de control son suficientes en el giro del negocio.

Usuarios, son los involucrados para que el marco de referencia funcione y que su toma de decisiones con respecto a tecnologías de información pueda aportar al negocio tanto responsablemente como controladamente todos los aspectos de la información

2.5.2. ITIL

Es un estándar de facto que fue introducido y distribuido por OGC en el Reino Unido en la que incluye todas las partes IT, según Stohlman Jr. & Brecher (2013), varios aspectos de seguridad se derivan del riesgo para empresas que mantienen proceso comerciales, los cuales dependen de mayor procesamiento de la información y cuya infraestructura es compleja y vulnerable a fallas e interrupciones. ITIL, refiere directamente al ciclo PDCA, donde enfatiza la necesidad de orientación del proceso, de la misma manera que una orientación del proceso.

2.5.3. LOPD

Ley española que de acuerdo al artículo 18.4 de la Constitución Española regula el uso y explotación de datos de carácter personal (Cutanda, 2010). Esta la ley establece obligaciones para el tratamiento de todo tipo de datos, y ya que es un documento legislativo no establece medidas ni técnicas para el cumplimiento de la misma.

Según Cutanda (2010), afirma que para aplicar la LOPS en necesario tener un entorno predeterminado y para cumplir con este estado hay que diseñar un plan de adaptación que garantice que se cumpla la ley, para ello es necesario seguir pasos previos a cada fase, como muestra la figura 1 se detallan los pasos de cumplimiento.

Figura 1. Pasos necesarios para el cumplimiento de la LOPD. Fuente: (Ingavélez, Hilera, Timbi, & Bengochea, 2016)

2.5.4. ISO/IEC 15408:2009

La norma facilita una guía de apoyo donde define un criterio estándar a usar como base para la valoración de las propiedades y características de seguridad sean estas de producto o sistemas de IT. La norma se constituye por tres fases las cuales se detallan a continuación.

Introducción y modelo general, según Sun, Gefei, Yajima, Miura y Shi (2012), esta fase describe antecedentes e ideas fundamentales en evaluación de seguridad donde se especifica un modelo de activos, modelo de contramedidas y modelo de evaluación en el aspecto de seguridad.

Seguridad funcional, es un catálogo de funcionalidades de seguridad que deben estar establecidas para el funcionamiento de las características de producto o sistema IT. (Sun et al., 2012)

Garantías de seguridad, es un catálogo de inspección de tareas para asegurar que las funciones de seguridad sean correctas. Con estos resultados definen los criterios de evaluación para siguientes evaluaciones (Sun et al., 2012)

2.5.5. ISO/IEC 21827:2008 SSE-CMM

Norma que describe características esenciales del proceso de ingeniería de seguridad de una organización, ISO /IEC 21827:2008 no prescribe un proceso o secuencia, sin embargo, captura prácticas observadas en la industria. En sus fases contempla al ciclo de vida, incluida actividades del desarrollo, mantenimiento y plan de mantenimiento, la norma establece interacciones con otras disciplinas y adicionalmente prueba la gestión de sistemas certificación acreditación y evaluación, según Kroll, Fontoura, Wagner, & Ornellas (2008), entrando en materia de seguridad todas las actividades deben ser aplicada para que el estudio de un programa de gestión de seguridad pueda facilitar el aumento de madurez de los procesos de ingeniería de seguridad dentro de la organización

2.5.6. ISO/IEC 27000

Proporciona una descripción general sobre los sistemas de gestión de seguridad de la información. Esta norma proporciona términos y definiciones que están relacionados con estándares de la ISMS. Según Stohlman Jr. & Brecher (2013), representa un consenso sobre las características como la calidad, la seguridad, y la fiabilidad que deben seguir aplicando durante largo periodos de tiempo. El objetivo del desarrollo de los estándares es vincular a individuos con empresas en la adquisición de productos y servicios

Los estándares de seguridad de la información consisten en un documento que contiene controles de seguridad aplicables a proyectos de seguridad de la información. Para comprender el propósito de los documentos es esencial entender que la normativa 27000 es el vocabulario estándar que provee una revisión básica de terminologías y definiciones.

Como muestra la figura 2 se establece que el punto de partida para el control está en la norma 27000 y de esta se desprenden certificaciones, guías y manuales correspondiente a la seguridad de la tecnología de la información.

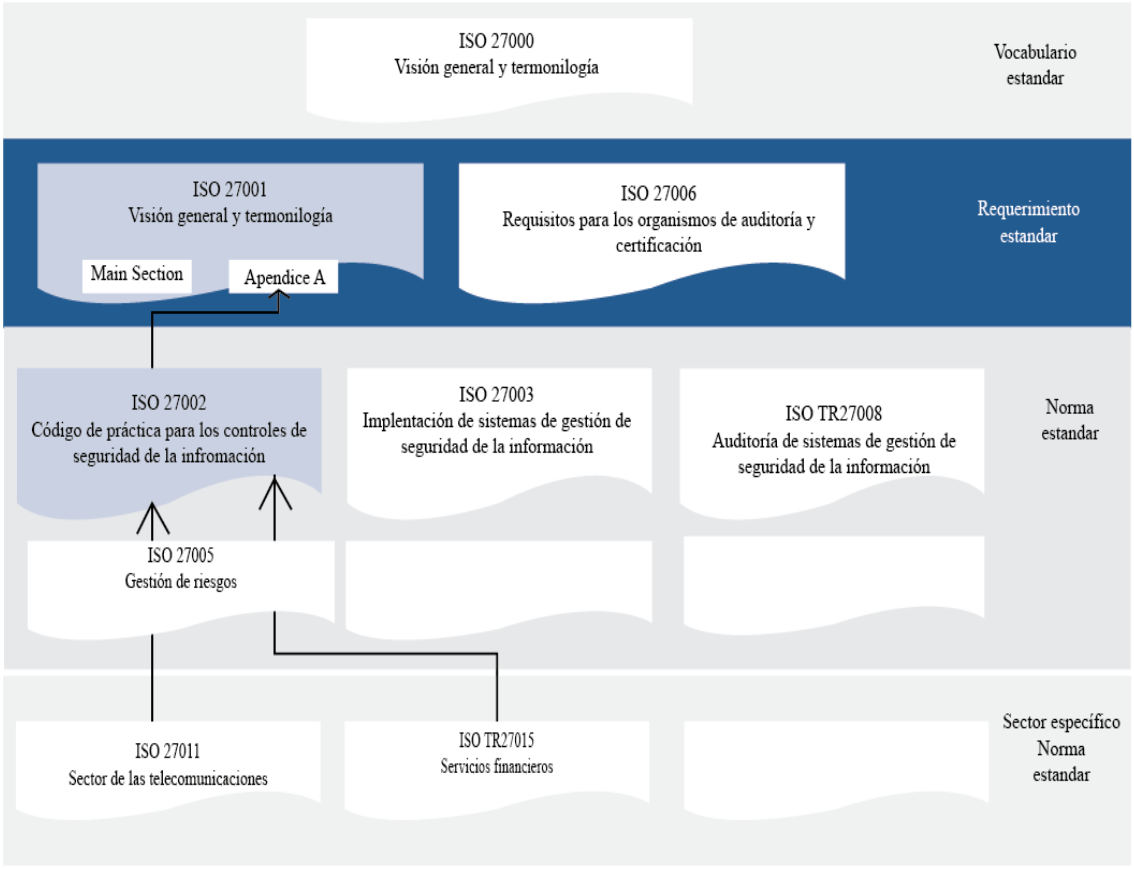

Figura 2. Resumen de la serie estándar 27000 Fuente: (Cuervo, 2017)

Con el fin de explicar la importancia de la seguridad de la información para las empresas, los riesgos que implican y la necesidad de haber apuntado y acordado controles en el marco de un SGSI, se exponen pasos necesarios para la identificación y evaluación de los riesgos de seguridad y sus campos de acción como muestra la figura 3

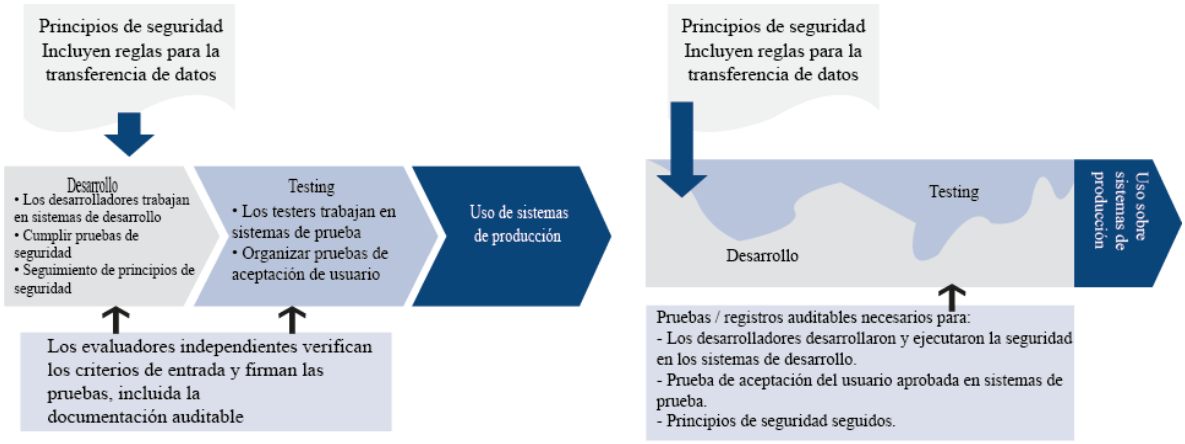

Figura 3. Como influye ISO 27001 en los procesos de desarrollo de software Modelo V / SCRUM Fuente: (Callejas-Cuervo et al., 2017)

2.5.6.1. ISO 27001

La norma describe los requisitos que un SGSI[1] deben ser cumplidos para lograr una certificación, esta norma se establece como un marco que está dirigido a empresas de todos los sectores sean estas pequeñas o grandes. Sin embargo, para Stohlman Jr. & Brecher (2013) existe alguna duda respecto a los requisitos SGSI para una PYME, ya que no existen medidas concretas para el cumplimiento de los requisitos y estas deben desarrollarse e implementarse sobre la base de la empresa en la cual se aplique.

La principal ventaja de la norma 27001 se basa en la implementación, operación, monitoreo y mejora continua de un SGSI orientado a procesos. Su enfoque se lo realiza alineado al ciclo PDCA[2] donde su alcance del SGSI debe ser definido para la planificación e implementación. Los riesgos deben identificar y evaluar tanto que se pueda definir objetivos de control para la información, tomando medidas adecuadas para proteger las operaciones. (Stohlman Jr. & Brecher, 2013).

[1] Sistema de gestión de la seguridad de la información
[2] Planificar, Hacer, Verificar y Actuar

2.5.6.1.1. Controles Anexo A

Los autores de la norma ISO 27001 (2013) proveen controles que permiten llegar a los objetivos de control para la aplicación de la seguridad en el desarrollo de productos de software, para el estudio se listan los siguiente controles.

Política de desarrollo seguro de software: Se deberían establecer y aplicar reglas para el desarrollo de software y sistemas dentro de la organización.(ISO/IEC_27001, 2013)

Procedimientos de control de cambios en los sistemas: En el ciclo de vida de desarrollo se deberían hacer uso de procedimientos formales de control de cambios.(ISO/IEC_27001, 2013)

Revisión técnica de las aplicaciones tras efectuar cambios en el sistema operativo: Las aplicaciones críticas para el negocio se deberían revisar y probar para garantizar que no se han generado impactos adversos en las operaciones o en la seguridad de la organización.(ISO/IEC_27001, 2013)

Restricciones a los cambios en los paquetes de software: Se deberían evitar modificaciones en los paquetes de software suministrados por terceros, limitándose a cambios realmente necesarios. Todos los cambios se deberían controlar estrictamente.

Uso de principios de ingeniería en protección de sistemas: Se deberían establecer, documentar, mantener y aplicar los principios de seguridad en ingeniería de sistemas para cualquier labor de implementación en el sistema de información.(ISO/IEC_27001, 2013)

Seguridad en entornos de desarrollo: Las organizaciones deberían establecer y proteger adecuadamente los entornos para las labores de desarrollo e integración de sistemas que abarcan todo el ciclo de vida de desarrollo del sistema.(ISO/IEC_27001, 2013)

Externalización del desarrollo de software: La organización debería supervisar y monitorear las actividades de desarrollo del sistema que se hayan externalizado.(ISO/IEC_27001, 2013)

Pruebas de funcionalidad durante el desarrollo de los sistemas: Se deberían realizar pruebas de funcionalidad en aspectos de seguridad durante las etapas del desarrollo.(ISO/IEC_27001, 2013)

Pruebas de aceptación: Se deberían establecer programas de prueba y criterios relacionados para la aceptación de nuevos sistemas de información, actualizaciones y/o nuevas versiones.(ISO/IEC_27001, 2013)

2.5.6.2. ISO 27002

2.5.6.2.1. Dominios de control

De acuerdo a los autores de la norma ISO 27002 (2013) mencionan que el objetivo de control de la normativa tiene como objetivo garantizar que la seguridad de la información se diseñe e implemente dentro del ciclo de vida de desarrollo de los sistemas de información. Es decir que para la implementación o desarrollo de software se deberían controlar estrictamente los entornos de desarrollo de proyectos y de soporte. Tomando esto como premisa se menciona que tanto los responsables de los sistemas de aplicaciones deben garantizar las propuestas de cambio en los sistemas tanto en la concepción del proyecto como en el ciclo de vida de desarrollo de sistemas en todas sus fases.

2.6. Ingeniería de software

Dentro del ámbito del desarrollo de software sus aplicaciones, sean estas técnicas o comerciales, facilitan las operaciones y la gestión de un negocio. Además del desarrollo de código fuente seguro, el software debe incluir la documentación y las características necesarias para que el aplicativo sea funcional en cualquier ambiente que esté preestablecido.

Según Castellaro, Romaniz, Ramos, Feck, & Gaspoz (2016) la ingeniería de software es la disciplina que abarca todos los aspectos del desarrollo de software, incluyendo las actividades de la ingeniería de requisitos, modelos de procesos y técnicas de estimación. Para Sommerville (2005), la ingeniería de software es una disciplina de la ingeniería que comprende todos los aspectos de la producción de software desde las etapas iniciales de la

especificación del sistema o recopilación de requerimientos hasta el mantenimiento del producto entregado.

La ingeniería de software tiene como misión la aplicación de un enfoque sistemático disciplinado, y cuantificable al desarrollo, operación y mantenimiento de software, donde la productividad está relacionada a una estimación de esfuerzo requerido durante la ejecución de proyectos. (Hernández, 2012).

Según Callejas, Alarcón, & Álvarez (2014)la ingeniería de software tiende a mejorar la eficiencia de las actividades dentro de los procesos razón por la cual el software de contar con criterios que garanticen su calidad..

2.6.1. Proyecto de software

Un proyecto es un esfuerzo temporal, destinado a crear un producto o prestar un servicio o un resultado único. De acuerdo a Rose (2013), un proyecto de software es la secuencia de fases en las cuales se incluye artefactos relacionados a su proceso que independientemente del modelo de ciclo de vida adoptado, involucra diversas áreas del conocimiento utilizadas en mayor o menor grado durante cada fase.

2.7. Ciclo de vida del software

Los autores de la normativa ISO 12007 (2008), define el ciclo de vida de un software como un proceso iterativo y secuencial compuesto de múltiples fases que culminan en la creación de un sistema de información independiente de su fin, estas fases suelen variar de una metodología a otra, pero como lo presenta Berzal (2018) las más distintivas son las siguientes:

- Planeación: En esta fase se define un plan de alto nivel preliminar sobre la ejecución prevista del proyecto de software, en donde es necesario conocer la factibilidad del proyecto con el objetivo de definir el alcance del problema y sus posibles soluciones, así mismo estimar recursos, beneficios y demás elementos necesarios para su ejecución.

- Recolección y análisis de requisitos: Esta fase implica analizar de forma detalla y minuciosa los requisitos del negocio provenientes del usuario final, en primera instancia se reúne requisitos funcionales, no funcionales y de seguridad, se crean los diagramas de procesos y luego se realiza un análisis estructurado de la información recopilada, siendo una de las primeras fases y de las más importantes. Es necesario que se destine el tiempo, energía y los recursos que sean necesarios para una acertada definición de las necesidades del cliente y de la funcionalidad del sistema.

- Diseño: En esta fase de describen en detalle las características y funcionalidades necesarias que satisfarán los requisitos funcionales del sistema que se implementará. Es durante esta fase que se consideraran la estructura de los componentes esenciales, el procesamiento y los procedimientos para alcanzar los objetivos de implementación.

- Desarrollo: Esta es la fase de codificación en donde se toman los diseños y requerimientos definidos en las fases anteriores y se transforman en el sistema, las dos actividades principales que se realizan en esta fase son el desarrollo de la infraestructura de TI que soportarán el sistema a implementar, y por otra parte la creación de la base de datos y los programas.

- Integración y pruebas: En esta fase se realiza la integración del sistema con los elementos con los que interactuará cuando esté en producción, como también se realizan pruebas a todos los programas y procedimientos desarrollados para determinar si el diseño propuesto cumple con el conjunto inicial de objetivos y alcances establecidos. Otra parte de esta fase es la verificación y validación, que ayudarán a garantizar la finalización exitosa del programa.

- Implementación y mantenimiento: Esta fase implica la instalación y configuración real del sistema y del entorno donde funcionaria; este paso pone en producción el sistema pasando del entorno desarrollo al entorno final. Además, del mantenimiento y actualizaciones requeridas a realizarse de forma regular, aquí los usuarios finales pueden ajustar el sistema, si lo desean, para aumentar el rendimiento, agregar nuevas capacidades o cumplir los requisitos adicionales del usuario.

2.8. Modelos de ciclo de vida

2.8.1. Modelo en cascada

En este modelo se presenta una secuencia de procesos ordenados, los cuales se desarrollan de manera lineal, por lo que pueden repetirse estados anteriores. Este modelo se caracteriza por lo siguiente: para continuar a la siguiente fase es necesario terminar la fase anterior, de igual manera se deben haber cumplido los objetivos propuestos en la fase preliminar, de acuerdo a la planificación adoptada se puede controlar las fechas de entrega, por lo que ayuda en el control de tiempos; por último, las personas involucradas tienen una retroalimentación del proyecto al término de cada fase, a continuación, se presentan las etapas más importantes de este modelo:

- Análisis de requisitos del sistema
- Análisis de requisitos del software
- Diseño preliminar
- Diseño detallado
- Codificación y pruebas
- Explotación y mantenimiento

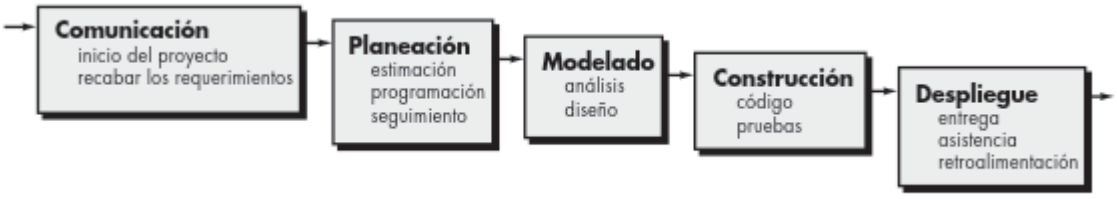

Figura 4. Diagrama modelo en cascada. Fuente: Pressman & Troya (2010)

Sin embargo, según Cova, Arrieta, & Aular De Duran (2008) este modelo presenta varios inconvenientes en cuanto a su funcionalidad, por ejemplo: la característica lineal es poco realista, puesto que en un proyecto existen iteraciones que no necesariamente tienen que ver con la etapa anterior, el sistema no se puede probar en el transcurso del proyecto,

sino que necesariamente se deben terminar todas las fases, por lo que el producto final se obtendrá una vez se hayan consumido todos los recursos, por lo que si se desea realizar un cambio es posible que el recurso económico no lo permita. Finalmente, debido a las características del modelo, los requisitos que se tengan para el proyecto no podrán cambiar una vez se dé inicio, por lo tanto, si surge algún requerimiento o cambio, no se podrá realizar.

Debido a lo expuesto, es necesario desarrollar una nueva versión de este modelo, la cual fue propuesto por (Lehman, Ramil, Wernick, Perry, & Turski, 1997), donde se basó en un diseño incremental, por lo que en cada fase se pueden añadir nuevos requisitos o funcionalidades, lo cual era una limitante en el modelo tradicional de cascada. Este modelo permite que los requisitos puedan irse puliendo a medida que transcurra el proyecto, por lo que pueden disminuir los costos y aumentar la eficacia.

2.8.2. Prototipado evolutivo

Según Mauro Callejas, Alarcón, & Álvarez (2014), este modelo se centra en ayudar a los usuarios a definir los requerimientos del proyecto, debido a que en muchas ocasiones no se tiene la certeza de lo que se desea realizar, también sirve como guía para los técnicos para determinar si las soluciones planeadas son viables. Los prototipos permiten tener una visión preliminar del producto, donde en un inicio las funcionalidades serán limitadas, pero incrementaran en cada etapa, por lo que las fases del ciclo de vida tradicional cambian, las cuales se detallan a continuación:

- Análisis de requisitos del sistema
- Análisis de requisitos de software
- Diseño, desarrollo e implementación del prototipo
- Prueba del prototipo
- Refinamiento iterativo del prototipo
- Refinamiento de las especificaciones del prototipo

- Diseño e implementación de sistema final

- Explotación y mantenimiento

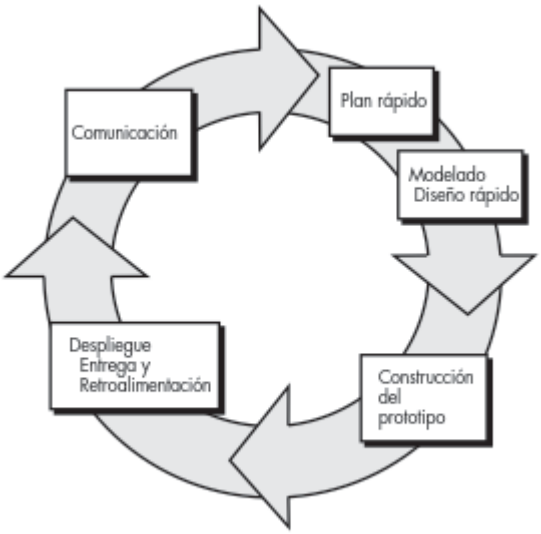

Figura 5. Diagrama paradigma de prototipos. Fuente:Peessman (2010)

2.8.3. Modelo en espiral de Boehm

Como se había mencionado con anterioridad, el modelo de cascada presenta varias limitaciones, por lo que (Boehm, 1988) desarrolló un modelo en espiral, el cual se ejecuta a partir de ciclos de desarrollo, por lo que en cada etapa va evolucionando. Según Gonzáles (2010), la estructura de este modelo es como su nombre lo indica como una espiral, puesto que está conformado por ciclos internos y externos y dimensiones radiales y angulares. En el ciclo interno se realiza un análisis por etapa y a su vez un prototipado, en el ciclo externo mantiene el modelo clásico, la dimensión radial consta de los costos y la dimensión angular del progreso obtenido en cada etapa.

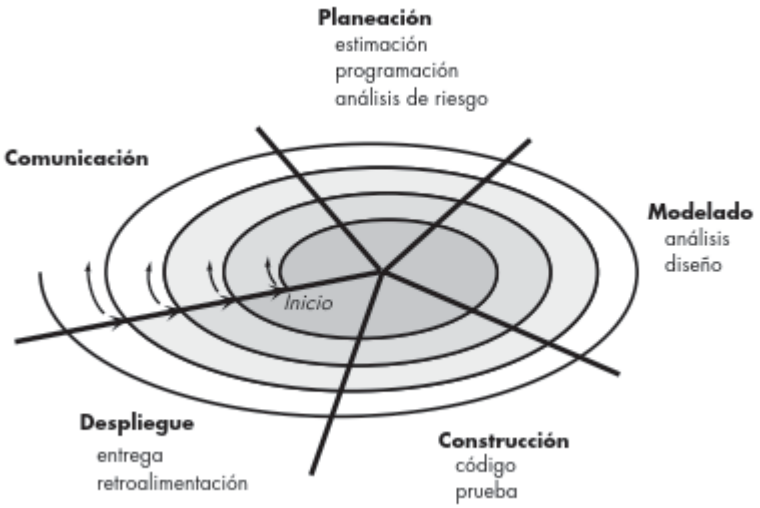

Figura 6. Modelo en espiral. Fuente: Pressman & Troya (2010)

Para comenzar con un ciclo es necesario identificar objetivos, alternativas y restricciones, para que se puedan elaborar alternativas de solución a los diferentes problemas, tomando en cuenta las restricciones, después de realizar este proceso se puede pasar al siguiente ciclo. Las salidas que se obtienen después de terminar cada ciclo es un análisis de riesgo, donde se puede identificar los posibles puntos donde es vulnerable el proyecto, con lo que se pueden buscar alternativas de solución para cada caso.

Según Cova, Arrieta & Aular de Duran (2008), este modelo de espiral presenta algunas ventajas sobre los modelos anteriores, por ejemplo: se tienen varias vías para cumplir los objetivos propuestos en cada ciclo, se obtiene un análisis de riesgo en cada etapa, por lo que se puede idear un plan de acción, el diseño es adaptable y aplicable a cualquier proyecto, puesto que no establece diferencias entre desarrollo de software y mantenimiento de sistema.

2.9 Modelos orientados a objetos

Los modelos tradicionales tienen un enfoque en el procedimiento, mientas que los modelos orientados a objetos el principal enfoque son los datos. Según Cataldi (2000) esta tecnología facilita acelerar el desarrollo de los sistemas de manera iterativa e incremental, con lo que se pueden reutilizar los componentes y disminuir costos, a continuación, se detallan algunos de estos modelos.

2.9.1 Modelo de agrupamiento o de Cluster

Este modelo está caracterizado por una fase de generalización, donde se combina también una fase de validación, corroborando su enfoque de agrupamiento. El agrupamiento consiste en juntar clases que tengan un objeto en común, con lo que se crean diferentes subciclos de vida, los cuales ayudan a disminuir el tiempo en el proyecto, es importante mencionar que cada uno de los subciclos creados dependen de los anteriores y su estructura es de forma ascendente, por lo que siempre se inicia desde las etapas básicas hasta el producto final (Verde & López, 2013).

2.9.2 Modelo de Fuente

El modelo de fuente fue desarrollado por (Henderson-Sellers & Edwards, 1990) con el fin de representar de manera gráfica las características de iteración y solapamiento con la que trabajan los modelos orientados a objetos.

2.9.3 Modelo de Remolino

Este modelo fue desarrollado por Rumbaugh, Blaha, Premerlani, & Lorensen (1991) como mejora al modelo de cascada, debido a que considera que este trabaja en una sola dimensión de iteración, por lo que limita el desarrollo. Este modelo propone diferentes dimensiones en cada etapa del proceso, las cuales se detallan a continuación:

- Amplitud, relacionada al tamaño del desarrollo;
- Profundidad, relacionado al nivel de detalle;
- Madurez, relacionado al grado de completitud;
- Alternativas, relacionadas a las diferentes soluciones a un determinado problema;
- Alcance, relacionado a los cambios en los requisitos.

Tomando en cuenta todas estas dimensiones propuestas, se puede obtener un modelo con diferentes ciclos, lo que ayuda a obtener un desarrollo menos lineal que los tradicionales.

2.10 Metodologías de desarrollo

Para el correcto desarrollo de cualquier proyecto es fundamental basarse en una metodología, es decir, seguir varios pasos o procedimientos consecutivos y sistemáticos de manera disciplinada, que permite obtener un resultado o producto deseado.

De manera general, en el desarrollo de proyectos de software, las metodologías permiten a los técnicos determinar los procedimientos, técnicas, herramientas y soporte documental que permitirá la obtención de un nuevo software. Según Cataldi (2000), las metodologías tienen tres requerimientos principales: obtener una mejor calidad en los productos, disminuir los costos y recursos, y lograr la estandarización de proceso para no afectar el mismo por algún cambio de personal.

Dentro de los objetivos se encuentran: elaborar un modelo sistemático, que permita un control del progreso en el desarrollo del proyecto, lograr que el cliente determine cuáles son los requerimientos reales del proyecto, lograr que los productos solicitados tengan una adecuada documentación y sean fáciles de mantener, facilitar la identificación oportuna y rápida de los cambios a realizar y la obtención de un sistema ágil y eficaz. (Cataldi, 2000)

La metodología de desarrollo es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con altas posibilidades de éxito, en el cual comprende actividades a seguir para idear, implementar y mantener un producto de software desde que nace la necesidad de realizar el producto hasta que se cumple el objetivo (Enríquez et al., 2017).

2.10.1 Metodologías estructuradas

Las metodologías estructuradas se enfocan en los procesos, flujogramas y estructura de los datos, es importante recalcar que se toma el modelo descendente, puesto que pasa de una visión general hasta llegar a algo específico. Existen diferentes tipos de estas metodologías. Por ejemplo, (Gane & Sarson, 1977), (DeMarco, 1979) y (Yourdon, 1975), las cuales siguen el diseño descendente, con lo que cada fase se descompone de manera específica y detallada.

La metodología de Gane & Sarson (1977) propone la construcción de un modelo que sea lógico y moderno; el método De Marco (1978) plantea desarrollar un modelo físico, lógico y moderno, tomando en cuenta costos y tiempos; el método de Yourdon (1975) es más práctico, por lo que elabora flujogramas del sistema, con el fin de conocer el proceso y determinar las acciones a realizar en cada etapa, para mejorar la calidad.

2.10.2 Metodologías ágiles

Las metodologías ágiles se caracterizan por ser iterativas e informales, debido a esto si se aplican a un proceso estricto, no lograrían cumplir con las expectativas, por lo que tendrían un resultado negativo; sin embargo, demuestran todo lo contrario, pues ofrecen los mismos resultados que otro tipo de métodos, con los mismos estándares de calidad, pero de una manera más ágil y segura, protegiendo datos, privacidad de usuarios y minimizando el uso de recursos (Cataldi, 2000).

Las metodologías agiles se caracterizan por lo siguiente: están fundamentadas en la creación de prácticas de producción de código, son fácilmente adaptables a cualquier cambio durante el proceso, las ideas son propuestas por el equipo, el proceso no es tan controlado, puesto que no se basa en tantos principios, los contratos establecidos son flexibles, el cliente es parte del desarrollo, lo cual es una gran ventaja para cumplir los requerimientos, los grupos de trabajo son pequeños, con un máximo de 10 personas, las cuales trabajan en un mismo sitio para facilitar la comunicación, la cantidad de recursos es mínima al igual que el recurso humano y ya no se pone tanto énfasis en la arquitectura del software (Letelier, Penadés, Canós, & Sánchez, 2012).

2.10.2.1 Extreme Programming (XP)

Es un tipo de metodología ágil que se centra en el recurso humano, es decir, en fortalecer las relaciones interpersonales para lograr éxito en el desarrollo de software, con lo que revitaliza el trabajo en equipo, de igual manera busca mejorar el aprendizaje de los desarrolladores para obtener un buen ambiente laborar. Se recomienda utilizar la metodología XP en proyectos donde los requerimientos son indefinidos y pueden cambiar en cualquier momento, por lo que existe un gran riesgo en la parte técnica. Como se mencionó con

anterioridad, este método al ser un tipo de metodología ágil, tiene roles designados dentro de cada proyecto, como por ejemplo: programador, cliente, tester, tracker, coach, consultor y gestor (Ghani & Yasin, 2013).

- Programador, es el encargado de elaborar el código del sistema.
- Cliente, determina los requerimientos del sistema, donde incluyen las expectativas del usuario y las pruebas que se deseen realizar para validar el funcionamiento.
- Tester, ayuda al cliente a definir los requerimientos, por lo que, al estar más familiarizado con este tema, es la persona encargada de ejecutar las pruebas de funcionalidad, de donde se obtendrán los resultados del proyecto, además indicará al proyecto si se lograron los resultados esperados o los cambios que se deseen realizar.
- El Tracker es la persona encargada de dar seguimiento y retroalimentar al equipo sobre el progreso del proyecto y las novedades que susciten, además controla que las tareas asignadas sean realizadas y en los tiempos establecidos.
- Coach, es la persona responsable de dar seguimiento a todo el proceso, es una guía para el proyecto, se encargará de que las practicas XP sean aplicadas correctamente, el Consultor es una persona externa al equipo, es a quien se le consultará cualquier duda que surja en el proceso, debido a que tiene el conocimiento necesario para resolverla.
- Gestor es el intermediario entre el cliente y los programadores, por lo que debe crear una atmosfera agradable de trabajo, básicamente se encarga de la coordinación.

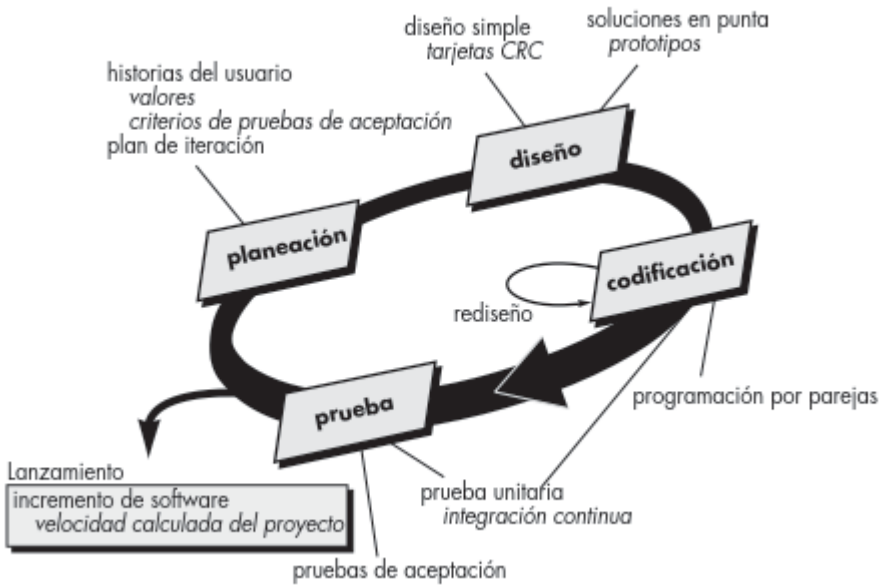

Figura 7. Diagrama programación extrema. Fuente: Pressman & Troya, (2010)

Según Letelier, Canós, Sánchez, & Penadés (2003), el proceso en el que se basa XP de manera general, consiste en los siguiente pasos:

- El cliente establece el rubro del proyecto que desea realizar
- El programador determina cual es el esfuerzo necesario que va a necesitar para cumplir los requerimientos del cliente.
- Conociendo la disponibilidad de los programadores el cliente decide que desea elaborar tomando en cuenta costos y tiempos.
- El programador construye el proyecto en base a las necesidades del cliente.
- Reinicia el ciclo, por lo que nuevamente el cliente definirá el valor del proyecto.

Es importante mencionar que, debido a los fundamentos de esta metodología, de fortalecer las relaciones interpersonales, no se debe forzar a los programadores a realizar trabajos que no estaban dentro de lo estimado y acordado, puesto que se perderá la calidad en el producto final, y es probable que no logren terminar tareas ya asignadas incrementando el tiempo estimado. Según Letelier, Penadés, Canós, & Sánchez (2009) el ciclo de vida de XP corresponde a las siguientes etapas: Exploración, Planificación de entrega, Iteraciones,

Producción, Mantenimiento y Muerte del proyecto, estas etapas son consideradas ideales dentro del proceso.

A criterio de Letelier et al.(2009) Uno de los objetivos más importantes que tiene la metodología XP se encuentra la disminución de los costos, debido a los cambios que se pueda generar a lo largo del proyecto; esto puede lograrse con el uso de tecnologías para el desarrollo de software y su disciplinada aplicación. Otra característica importante es la constante comunicación entre los usuarios y clientes, con lo que se determinará un adecuado tiempo de entrega del producto; para lograr que el tiempo establecido se cumpla XP plantea que se realicen pequeños entregables en cada etapa, con lo que se puede ir evaluando la funcionalidad del software, realizando los cambios necesarios en el momento y no al finalizar el proceso

En el ámbito de la programación XP tiene una característica singular, la producción de código debe ser elaborada en parejas, lo que ayuda a disminuir los errores que podrían suscitarse debido a que se tiene una doble verificación.

Según Rindell, Hyrynsalmi, & Leppanen (2015), en relación a la codificación es importante la propiedad colectiva del mismo, es decir, que cualquier programador podrá cambiar cualquier parte del código cuando sea necesario. Dentro de las necesidades que tiene la metodología XP se encuentra que las jornadas de trabajo no deben sobrepasar las 40h semanales, si se trabajan horas extras significa que no se está cumpliendo el tiempo planificado, por lo que se deben reevaluar las tareas, además que un exceso de trabajo desmotiva a los trabajadores. El cliente debe estar siempre disponible para los programadores, es un factor clave para el éxito del proyecto, debido a que se transmitirán adecuadamente los requerimientos del usuario y se podrán resolver todas las dudas necesarias; esta comunicación debe ser de preferencia oral, debido a que es más eficiente que la escrita.

Si se aplican todos los requerimientos y se siguen todos los procesos de esta metodología se logrará obtener un software eficaz, ágil, rápido y con una inversión menor. Según Rindell et al. (2015) la aplicación de XP debe ser respecto a la perspectiva del negocio, el recurso humano y el trabajo en equipo.

2.10.2.2 Scrum

Esta metodología fue desarrollada por Schwaber & Beedle (2002), la cual está dirigida a proyectos que requieren un cambio constante de los requisitos. Según Becerra (2018) las principales características de esta metodología se basan en que el desarrollo de software se realiza mediante iteraciones denominadas *sprints*[3], con un tiempo de duración de 30 días, el resultado de estos *sprints* corresponde a los resultados paulatinos que se entregan durante la implementación. La segunda característica corresponde a las reuniones que se realicen a lo largo del proyecto, de preferencia se recomienda una reunión diaria de 15 minutos donde el equipo de desarrollo pueda coordinarse e integrarse.

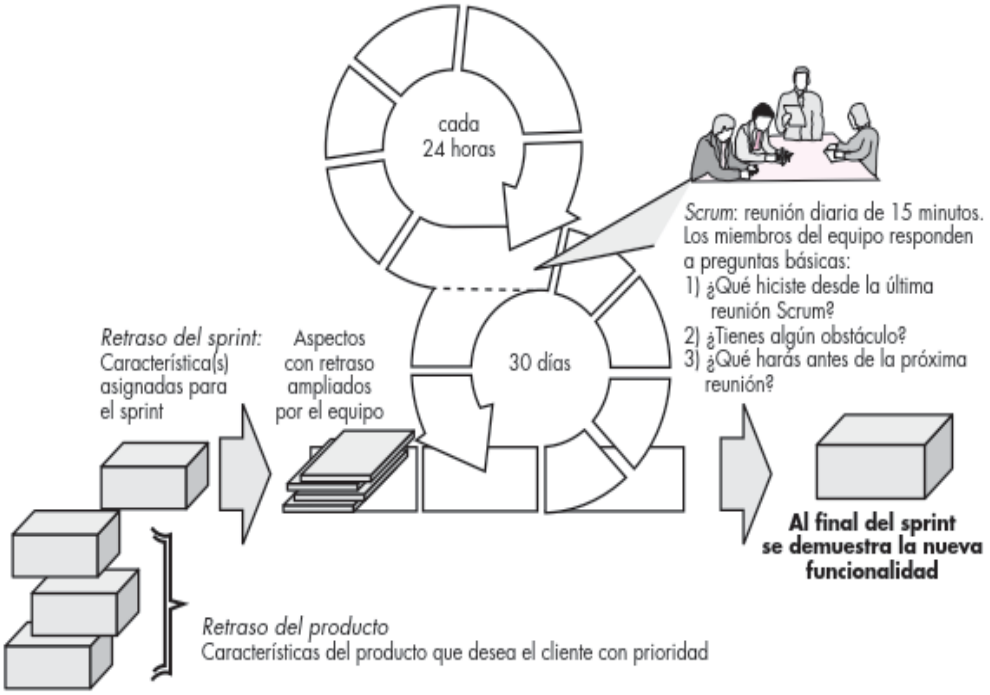

Figura 8. Flujo de trabajo scrum. Fuente: Pressman & Troya, (2010)

[3] Nombre que va a recibir cada uno de los ciclos o iteraciones dentro de un proyecto Scrum

2.10.2.3 Cristal methodologies

Según Pressman & Troya (2010) es un conjunto de metodologías que se caracterizan por priorizar el recurso humano, equipo de trabajo, y la reducción considerada de artefactos producidos. Es una interrelación entre invención y comunicación, donde los recursos se encuentran limitados. A criterio de Pressman & Troya (2010) los recursos deben ser empleados en la conformación del equipo de trabajo, ya que de acuerdo a su cantidad se establecen niveles de operación, para esto los equipos deben mejorar sus habilidades y destrezas como programadores en el ámbito técnico, pero también deben fortalecer el trabajo en equipo de manera administrativa, para visualizar sus características se pueden clasificar mediante colores en función de la cantidad de personas en el equipo, como se puede apreciar en la tabla 1.

Recurso	Cantidad de personas				
	Claro	Amarillo	Naranja	Rojo	Marrón
Vida(L)	L6	L20	L40	L80	L200
Dinero Esencial(E)	E6	E20	E40	E80	E200
Dinero discrecional (D)	D6	D30	D40	D80	D200
Comodidad (C)	C6	C20	C40	C80	C200
	1 a 6	7 a 20	21 a 40	41 a 80	81 a 200

Tabla 1. Diagrama de calor metodología de cristal Fuente: https://univertis.com/general-en/introduction-to-agile-software-development/

2.10.2.4 Dynamic systems development method (DSDM)

Según Pressman & Troya (2010), las características de esta metodología se basan en que comprende un proceso iterativo incremental adicionando al equipo de trabajo y usuario, los cuales trabajan juntos entre sí. Este método consta de las siguientes fases: estudio de la factibilidad del proyecto, estudio del negocio, modelado funcional, diseño, construcción e implementación.

2.10.2.5 Adaptive software development (ASD)

Pressman & Troya (2010) mencionan que el modelo creado por Highsmith (2000), se caracteriza por ser iterativo, se orienta a los componentes del software en lugar de las tareas

por etapa, como lo hacen otros modelos, y es adaptable a los cambios. El ciclo de vida que propone este método consta de tres fases: especulación, colaboración y aprendizaje.

- Especulación inicia el proyecto y es ahí donde se plantean los requerimientos del proyecto.
- Colaboración, se elaboran las características definidas del software.
- Aprendizaje, se evalúa la calidad del producto y posterior a eso se realizará la entrega final al cliente.

Es importante recalcar que este método enfatiza en la revisión de componentes, con lo que se pueden detectar errores, solucionarlos y aprender de ellos, para tomarlos en cuenta al iniciar el siguiente ciclo.

Figura 9. Diagrama desarrollo adaptativo. Fuente: Pressman & Troya (2010)

2.10.2.6 Feature-driven development (FDD)

Esta metodología se basa en un procedimiento iterativo en cinco etapas, además la iteración es una de sus características, las cuales son cortas (alrededor de dos semanas). Las fases en las que se centra son las de diseño e implementación, basándose en los requerimientos definidos para el software (Ambler, 2004).

2.10.2.7 Lean development (LD)

La metodología propuesta por Bob Charette's (1990) se basa en permitir que los riesgos generados en un proyecto se conviertan en oportunidades de mejora en la productividad, con lo cual este método consta con un mecanismo de implementación de soluciones a dichos riesgos (DeMarco, 1979). Por ejemplo, los cambios generados se consideran riesgos y su eliminación conduce a la eficiencia global del proceso de desarrollo, esto a su vez se transforma en una reducción de tiempo y costo del proyecto (Letelier et al., 2003).

2.11 Metodologías de desarrollo de software seguro

Para Diaz (2015) el desarrollo de software seguro es importante considerar el uso de un buen proceso de desarrollo, el cual permita resistir y mitigar daños o ataques que se presenten durante su ejecución de forma efectiva. Muchos de los problemas de seguridad pueden ser mitigados si los desarrolladores establecen reglas de seguridad durante todo el ciclo de vida del proyecto. A continuación, se presentan diferentes tipos de metodologías para el desarrollo de software seguro.

2.11.1 Correctness by Construction (CbyC)

Según Brito (2013) existen diferentes metodologías que se pueden aplicar para desarrollar un software seguro, una de ellas es Correctness by Construction (CbyC), el cual posee un nivel de calidad estricto, por lo que busca obtener el mínimo de defectos y una adecuada adaptación a los cambios; mediante los siguientes fundamentos: dificultad de ingresar errores y en el caso de que ingresen que sean eliminados inmediatamente. Con esta metodología se puede lograr que el software sea seguro desde su inicio hasta el producto final, por los estándares utilizados, la solidez de todas las etapas del proceso, la agilidad para detectar problemas y la constancia de los datos obtenidos; además permite obtener retroalimentación constante por su característica de desarrollo incremental.

2.11.2 Security Development Lifecycle (SDL)

Según Brito (2013) la metodología SDL fue desarrollada por Microsoft en el año 2004 con el fin de lograr un control de la seguridad en desarrollo de software, debido a que se concentra en todas las etapas del proceso, por lo que es posible detectar fallas en el código, proteger todo el código de posibles atacantes, generar una mejora continua en cada etapa, remover códigos y funciones antiguas y actualizarlo constantemente. Todo esto es posible lograr si se utilizan diferentes herramientas como: modelado de amenazas, gestión de riesgos o codificación segura.

Brito (2013) explica que existen diferentes tipos de metodologías SDL, las cuales se diferencian en el tipo de software que se desee realizar. La versión más estricta de SDL tiene un enfoque más apropiado para proyectos grandes y que no sean vulnerables a los cambios que se puedan realizar durante el proceso, mientras que la versión ágil está enfocada en el desarrollo web. La metodología SDL se fundamenta en tomar en cuenta todos los focos de vulnerabilidad desde las fases preliminares hasta el producto final, por lo que analiza cada fase del proceso, las cuales se detallan a continuación.

Según Brito (2013) esta metodología cuenta con siete etapas:

- Entrenamiento, busca capacitar continuamente a todo el personal involucrado en el proyecto sobre medidas de seguridad y privacidad, específicamente a los técnicos, los cuales están directamente relacionados, en este caso el enfoque debe ser para el modelo y evaluación de riesgos, así como también código SQL.

- Requerimientos, se consultará con un especialista sobre el plan de trabajo y los objetivos que se desean lograr en torno a la seguridad, además se presentaran los lugares donde son necesarias medidas de seguridad para que nada pase desapercibido.

- Diseño, se establecerá la arquitectura del software, es decir, se realizará el modelo de cómo se elaborará el software y se definirán las áreas que requieren minimizar los ataques, por lo que a la par se determinarán las amenazas en cada punto.

- Implementación, se tomarán en cuenta las áreas donde se detectaron las amenazas, por lo que el código que se elaborará intentará mitigar cada uno de los peligros, basándose en normativas, es importante mencionar que se realizarán varias pruebas para evaluar la eficacia y agilidad de respuesta frente a algún tipo de ataque

- Verificación, se realizarán pruebas más exhaustivas del proyecto, debido a que se encuentra terminado.

- Lanzamiento, se someten a varias pruebas con el cliente que dura hasta seis meses, con el fin de detectar alguna falla en el código, con lo cual se garantiza la eficacia.

- Respuesta, se debe tener en cuenta que siempre existirá un mínimo de error, por lo que el software debe estar preparado para detectar y solucionar cualquier tipo de ataque.

2.11.3 Cigital touchpoints

Es un conjunto de mejores prácticas para el desarrollo seguro, aparece en el 2004 en la revista IEEE Security & privacy. Según McGraw (2006), esta metodología se basa en tres pilares de seguridad, los cuales son el riesgo aplicado sobre la gestión, puntos de contacto de seguridad de software y el conocimiento. Davis, Humprey, Redwine, Zibulsk y McGraw (2004), sugieren que al aplicar estos tres pilares de manera gradual y evolutiva en una medida equitativa pueden dar como resultado un programa razonable y rentable con el aspecto de seguridad de software. Los puntos de contacto dentro de la metodología touchpoint proporcionan una combinación de actividades de sombrero negro [4]y sombrero blanco [5]los cuales son necesarios para obtener resultados objetivos.

Según Tiirik (2004), no todos los puntos de contacto deben adoptarse para empezar a crear seguridad en el desarrollo de software, sin embargo establece que es recomendable hacerlos todos, también menciona que los puntos de control de seguridad de software son mejor aplicados por personas ajenas en el diseño original y en la implementación del sistema.

[4] Son ataques, explotaciones a software comúnmente llamadas pruebas de penetración
[5] Están relacionadas con el diseño, los controles y la funcionalidad

La aplicación de los puntos de contacto depende exclusivamente de la naturaleza en la cual se desarrolle, sin embargo, Tiriik (2004) menciona que existen orden de efectividad las cuales son: revisión de código, análisis de riesgos arquitectónicos, pruebas de penetración, pruebas de seguridad basada en riesgos, casos de abuso, requisitos de seguridad y operaciones de seguridad.

2.11.4 Common criteria

La metodología Common Citeria está orientada al desarrollo, evaluación y adquisición de productos de Tecnologías de Información que tengan relación con la seguridad; además, permite comparar resultados de evaluaciones de seguridad individuales, debido a esto se logra conformar un conjunto de requisitos en común. Para dar inicio a este proceso, en primer lugar, se verifica el cumplimiento de los objetivos propuestos, tomando en cuenta los peligros detectados y la funcionalidad; además, esta metodología permite diseñar procesos con el mínimo de riesgo, máxima seguridad y confiabilidad en cuanto a la garantía (Keramati & Mirian-Hosseinabadi, 2008)

2.11.5 Comprehensive lightweight application security process (CLASP)

CLASP es un proceso ligero para construir software seguro, el cual incluye 24 actividades de alto nivel, que se adaptan al proceso de desarrollo. A criterio de Wouter (2009) la característica principal es la seguridad en el centro del escenario donde se define y conciben desde una perspectiva teórica de seguridad y por esa razón el conjunto de actividades es amplia.

Según De Win, Scandariato, Buyens, Grégoire y Joosen (2009), esta metodología es un tanto limitada ya que define un conjunto de elementos independientes llamadas actividades que luego deben integrarse en el proceso de desarrollo y su ejecución es abierta a la flexibilidad, en la cual se definen mapas de ruta que guían en la combinación de actividades de forma coherente y ordenada.

2.11.6 Tsp-secure (TSP-S)

Esta metodología es desarrollada por la Carnegie Mellon Univeristy Software Engineering Institute (SEI), a criterio de Davis, Humprey, Redwine, Zibulsk y McGraw (2004) es un proceso operativo para el uso de equipos de desarrollo para la implementación de software seguro. Su principal virtud es que antes del proceso de análisis de requerimientos el equipo de desarrollo recibe capacitación en aspectos de seguridad donde incluyen causas comunes de vulnerabilidades y métodos orientados a la seguridad, los métodos de diseño son orientados a la seguridad y los métodos de implementación son orientado a listas de verificación.

2.12 Calidad en ingeniería de software

El aseguramiento de la calidad del software (ACS) es un proceso de tareas específicas de aseguramiento y control de la calidad donde se incluyen revisiones técnicas y una estrategia de pruebas relacionadas entre sí. Según Davis, Humprey, Redwine, Zibulsk y McGraw (2009), las prácticas eficaces de ingeniería de software entiende por un control de todos los productos del trabajo de software y de los cambios que durante el procedimiento de implementación, el cual garantiza el cumplimiento de estándares antes, durante y después del desarrollo. La meta de la calidad de software se centra en establecer un conjunto de objetivos o atributos donde garanticen la calidad en las fases del ciclo de vida desarrollo del producto.

En la tabla 2 se observa cómo está distribuida las metas y las posibles métricas que puede aplicar a cada proyecto.

Meta	Atributo	Métrica
Calidad de los requerimientos	Ambigüedad	Número de modificadores ambiguos (por ejemplo, muchos, grande, amigable, etc.)
	Completitud	Número de TBA y TBD
	Comprensibilidad	Número de secciones y subsecciones
	Volatilidad	Número de cambios por requerimiento
		Tiempo (por actividad) cuando se solicita un cambio
	Trazabilidad	Número de requerimientos no trazables hasta el diseño o código
	Claridad del modelo	Número de modelos UML
		Número de páginas descriptivas por modelo
		Número de errores de UML
Calidad del diseño	Integridad arquitectónica	Existencia del modelo arquitectónico
	Completitud de componentes	Número de componentes que se siguen hasta el modelo arquitectónico
		Complejidad del diseño del procedimiento
	Complejidad de la interfaz	Número promedio de pasos para llegar a una función o contenido normal
		Distribución apropiada
	Patrones	Número de patrones utilizados
Calidad del código	Complejidad	Complejidad ciclomática
	Facilidad de mantenimiento	Factores de diseño
	Comprensibilidad	Porcentaje de comentarios internos
		Convenciones variables de nomenclatura
	Reusabilidad	Porcentaje de componentes reutilizados
	Documentación	Índice de legibilidad

Meta	Atributo	Métrica
Eficacia del control de calidad	Asignación de recursos	Porcentaje de personal por hora y por actividad
	Tasa de finalización	Tiempo de terminación real versus lo planeado
	Eficacia de la revisión	
	Eficacia de las pruebas	Número de errores de importancia crítica encontrados
		Esfuerzo requerido para corregir un error
		Origen del error

Tabla 2. Metas atributos y métricas de la calidad del software. Fuente: Pressman & Troya (2010)

2.12.1 Métricas del software

Según Gonzales (2001), métrica se define como una medida cuantitativa del grado en que un sistema, componente o proceso posee un atributo dado, en este sentido una métrica de software se refiere a la aplicación continua de mediciones basadas en técnicas para el proceso de desarrollo de software y sus productos para informar en el instante, de esta manera se podrá lograr una mejora continua del proceso, con toda esta información se facilita la toma de decisiones en el ámbito técnico.

Las métricas colaboran a una correcta evaluación de modelos de análisis y diseño, donde se determinará la seguridad de los procedimientos de diseño y código de fuente, con lo que las pruebas serán más efectivas. El procedimiento de medición que se utiliza consta de cinco fases, las cuales detallaremos a continuación:

- Formulación, donde se tomarán medidas y métricas, las cuales serán una representación del software.

- Colección, donde se debe implementar un sistema de recolección de datos para obtener las métricas necesarias.

- Análisis, se procede al cálculo de las métricas mediante la aplicación de herramientas matemáticas.

- Interpretación, es donde se evaluarán los resultados obtenidos de las métricas, con el fin de tener indicadores de calidad del software.

- Retroalimentación, se realizarán sugerencias o recomendaciones sobre los resultados obtenidos sobre las interpretaciones emitidas sobre el software.

2.12.1.1 Métricas del modelo de análisis

En esta fase se obtendrán los requisitos y se establecerá el fundamento para el diseño. Y es por eso que se desea una visión interna a la calidad del modelo de análisis. Sin embargo, se entiende que, en el análisis y especificación, es posible adaptar métricas obtenidas para la aplicación de un proyecto, es decir estas métricas examinan el modelo de análisis con el fin de predecir el tamaño del sistema resultante, en donde el tamaño y la complejidad del diseño estén directamente relacionadas. Es por eso que se verán en las siguientes secciones las métricas orientadas a la función, la métrica bang[6] y las métricas de la calidad de especificación.

2.12.1.1.1 Métricas basadas en la función

Según Pressman & Troya (2010), las métricas basada en la función se puede usar como medio para predecir el tamaño de un sistema que se va a obtener de un modelo de análisis. Estos datos históricos proporcionan al administrador del proyecto una importante información de planificación basada en el modelo de análisis en lugar de en estimaciones preliminares.

2.12.1.1.2 Métricas de la Calidad de especificación

Existe una lista de características para poder valorar la calidad del modelo de análisis y la correspondiente especificación de requisitos: Especificidad, corrección, compleción, comprensión, capacidad de verificación, consistencia externa e interna, capacidad de logro, concisión, trazabilidad, capacidad de modificación, exactitud y capacidad de reutilización. Aunque muchas de las características anteriores pueden ser de naturaleza cuantitativa,

[6] Métrica que permite estimar el tamaño del producto de software desde el punto de vista del usuario e independientemente de su implementación

Pressman & Troya (2010), sugieren que todas puedan representarse usando una o más métricas, no obstante, incorpora una métrica global completa, la cual se debe considerar el grado de validación de los requisitos y medir el cumplimiento de los mismos

$$Q3 = nc^{7} / (nc + nnv^{8})(1)$$

2.12.1.1.3 Métrica del modelo del diseño

Las métricas para software, como otras, no son perfectas; Gonzales (2010) afirma, que se necesita más experimentación hasta que se puedan emplear bien las métricas de diseño.. A continuación, se mostrarán algunas de las métricas de diseño más comunes, las cuales proporcionan al diseñador una mejor visión interna de los requerimientos cumplir.

2.12.1.1.4 Métricas de código de fuente

Según Gonzales (2010), la ciencia software propuso las primeras 'leyes' analíticas para el software de computadora. La ciencia del software asigna leyes cuantitativas al desarrollo del software de computadora. La teoría de Halstead (1977) se obtiene de un supuesto fundamental (Pressman & Troya, 2010) El cerebro humano sigue algoritmos rígidos sin tener en cuenta que lo está realizando.

En palabras de Pressman & Troya (2010) la ciencia del software usa un conjunto de medidas primitivas que pueden obtenerse una vez que se ha generado o estimado el código después de completar el diseño. Estas medidas se listan a continuación. n1: el número de operadores diferentes que aparecen en el programa, n2: el número de operandos diferentes que aparecen en el programa, N1: el número total de veces que aparece el operador, N2: el número total de veces que aparece el operando.

Halstead (1977) usa las medidas primitivas para desarrollar expresiones para la longitud global del programa; volumen mínimo potencial para un algoritmo; el volumen

[7] NC número de requisitos que se han validados como correctos
[8] NNV número de requisitos que no se han validado todavía

real[9]; el nivel del programa[10]; nivel del lenguaje[11]; y otras características tales como esfuerzo de desarrollo, tiempo de desarrollo e incluso el número esperado de fallos en el software. (Halstead, 1977), expone que la longitud N se puede estimar como:

$$N = n1 \log2 n1 + n2 \log2 n2$$

El volumen de programa se puede definir como:

$$V = N \log2 (n1 + n2)$$

Se debería tener en cuenta que V variará con el lenguaje de programación y representa el volumen de información teóricamente debe existir un volumen mínimo para un algoritmo. (Halstead, 1977), también define una relación de volumen L como la relación volumen de la forma más compacta de un programa respecto al volumen real del programa. Por tanto, L, debe ser siempre menor de uno. En términos de medidas primitivas, la relación de volumen se puede expresar como:

$$L = 2/n1*n2/N2 \quad (4.35)$$

(Halstead, 1977), propuso que todos los lenguajes pueden categorizarse por nivel de lenguaje, adicionalmente, creó una teoría que suponía que el nivel de lenguaje es constante para un lenguaje dado, sin embargo, Pressman & Troya (2010), indica que el nivel de lenguaje es función tanto del lenguaje como del programador.

2.12.1.1.5 Métricas para pruebas

En palabras de Pressman & Troya (2010) las métricas propuestas se concentran en el proceso de pruebas. En general, los responsables de las pruebas deben fiarse del análisis, diseño y código para que les guíen en el diseño y ejecución de los casos de prueba.

Las métricas basadas en la función pueden emplearse para predecir el esfuerzo global de las pruebas. El número de primitivas funcionales (Pfu), elementos de datos, (ED), objetos

[9] Número de bits requeridos para especificar un programa
[10] Medida de la complejidad del software
[11] Constante para un lenguaje dado

(OB), relaciones (RE), estados (ES) y transiciones (TR) pueden emplearse para predecir el número y tipos de pruebas de la caja negra y blanca del software o de forma general en pruebas de funcionamiento. Por ejemplo, el número de pruebas asociadas con la interfaz hombre-máquina se puede estimar examinando el número de transiciones (TR) contenidas en la representación estado transición del IHM[12] y evaluando las pruebas requeridas para ejecutar cada transición, el número de objetos de datos (OB) que se mueven a través de la interfaz y el número de elementos de datos que se introducen o salen.(Pressman & Troya, 2010)

El esfuerzo de las pruebas también se puede estimar usando métricas obtenidas de medidas de Halstead (1977). Usando la definición del volumen de un programa V, y nivel de programa, NP, el esfuerzo de la ciencia del software, puede calcularse como;

NP=1 / [(n1 / 2) * (N2 / n2)] -> e = V / NP

A medida que se van haciendo las pruebas tres medidas diferentes proporcionan una indicación de la complexión de las pruebas. Una medida de la amplitud de las pruebas proporciona una indicación de cuantos se han probado. Según Calabrese, Muñoz, Pasini, Esponda y Boracchia (2017), mencionan que se puede calcular una estimación razonablemente exacta del número de caminos básicos sumando la complejidad ciclomática de todos los módulos del programa. Finalmente, a medida que se van haciendo las pruebas y se recogen los datos de los errores, se pueden emplear los perfiles de fallos para dar prioridad y categorizar los errores descubiertos. La prioridad indica la severidad del problema. Las categorías de los fallos proporcionan una descripción de un error, de manera que se puedan llevar a cabo análisis estadísticos de errores.

2.12.1.1.6 Métricas de mantenimiento

Se han propuesto métricas diseñadas explícitamente para actividades de mantenimiento. Pressman & Troya (2010), sugiere un índice de madurez del software (IMS) que proporciona una indicación de la estabilidad de un producto de software (basada en los cambios que ocurren con cada versión del producto). Se determina la siguiente información:

[12] Interfaz hombre máquina

Mr = número de módulos en la versión actual Fc = número de módulos en la versión actual que se han cambiado Fa = número de módulos en la versión actual que se han añadido Fd = número de módulos de la versión anterior que se han borrado en la versión actual. El índice de madurez del software se calcula de la siguiente manera:

IMS = [Mr – (Fa + Fc + Fd)]/ Mr.

A medida que el IMS13 se aproxima a 1.0 el producto se empieza a estabilizar. El IMS puede emplearse también como métrica para la planificación de las actividades de mantenimiento del software. El tiempo medio para producir una versión de un producto software puede correlacionarse con el IMS desarrollándose modelos empíricos para el mantenimiento.

2.13 Estado del Arte

Dentro del desarrollo de proyectos, la seguridad es un tema importante para lograr que todo el proceso se desempeñe de forma continua y correcta durante todo su desarrollo. Existen muchas vulnerabilidades que son introducidas involuntariamente en el software durante el diseño y desarrollo. Por tal motivo, es necesaria la aplicación seguridad en todo el proceso del ciclo de vida del proyecto, tomando en cuenta normativas, regulaciones, estándares o herramientas de seguridad para mejorar las prácticas de ingeniería de software, donde se presta atención sustancialmente a los defectos generales de especificación diseño e implementación, así como vulnerabilidades de software. A continuación, en la tabla 3 se detalla una compilación de diferentes trabajos de investigación sobre el uso de seguridad dentro en el ciclo de vida del desarrollo del software SDSL.

Metodología	Autores	Resultados
Correctness by construction (cbyc)	(Chapman, 2005)	Los defectos se eliminan tempranamente
	(Brito, 2013)	Los cambios son baratos. Las pruebas se convierten en una validación del software. La seguridad se produce como un subproceso del proceso Las primeras iteraciones producen software con funciones útiles.

[13] Sistema de gestión de la información

Metodología	Autores	Resultados
Security development lifecycle (SDL)	(Howard, 2004)	Puede usarse para procesar información personal o confidencial.
	(Brito, 2013)	Puede ser usada en cualquier empresa u otra organización, incluida la academia, el gobierno o las organizaciones sin fines de lucro.
	(Lipner, 2004)	Puede conectarse a Internet o estar conectado a cualquier entorno en red.
		(Wouter, 2009)
Digital touchpoints	(Wouter, 2009)	Sus tres pilares de la seguridad del software son la gestión de riesgos aplicados, los puntos de contacto de seguridad del software y el conocimiento.
	(Tiirik, 2004)	Los puntos de contacto de seguridad de software se aplican mejor por personas que no participan en el diseño original y la implementación del sistema.
		(McGraw, 2006)
Comprehensive lightweight application security process (CLASP)	(Wouter, 2009)	Es un método sistemático, aunque se pueden mejorar varias actividades, por ejemplo, la identificación de recursos o casos de uso indebido a nivel arquitectónico.
	(Kara, 2012)	La ejecución del proceso de identificación de requisitos puede tomar demasiado tiempo.
Tsp-secure	(Davis et al., 2004)	Capacitación adicional a los desarrolladores de software en seguridad, por ejemplo, vulnerabilidades de seguridad, métodos de diseño orientados a la seguridad, métodos de implementación conscientes de la seguridad y métodos de prueba de seguridad.

Tabla 3. Metas atributos y métricas de la calidad del software. Fuente: Pressman & Troya (2010)

Las metodologías de aseguramiento al modelo del ciclo de vida son diversas, sin embargo, existen puntos de control que coinciden en sus criterios en común, como muestra la tabla 4.

Atributo/ Metodología	**CbyC**	**SDL**	**Touchpoints**	**CLASP**	**TSP-S**
Utilización de métodos formales	x				
Desarrollo iterativo	x		x	x	x
Entrenamiento en temas de seguridad		x	x	x	x
Establece requerimientos de seguridad	x	x	x	x	x
Estudia la trazabilidad de los requerimientos	x			x	
Asistencia personal especializado en seguridad		x	x	x	x
Realiza modelado de amenazas		x	x	x	

Atributo/ Metodología	CbyC	SDL	Touchpoints	CLASP	TSP-S
Análisis estático	x	x			x
Análisis dinámico		x	x	x	
Fuzz testing		x			
Revisión de código	x	x	x	x	x
Desarrolla un plan en caso de incidentes de seguridad		x	x		x

Tabla 4. Criterios comunes en metodologías SDSL

Como parte de un ejemplo las metodologías ofrecen métricas con las cuales se puede obtener el riesgo s través de todo su proceso, como se puede apreciar en la tabla 5 utilizando la metodología CbyC, sus métricas son LOCs[14] respecto a la productividad del día y el número de defectos por cada mil líneas de código. Con estos datos se refleja que a medida que pasan los años la madurez en la ejecución de proyectos mejora la calidad y productividad respecto a años anteriores.

Project	Year	Tamaño (loc)	Productividad del ciclo de vida completo (loc por día)	Defectos (por kloc)
CDIS[15]	1992	197000	12,7	0,75
SHOLIS[16]	1997	27000	7	0,22
MULTOS CA[17]	1999	100000	28	0,04
A[18]	2001	39000	11	0,005
NSA[19]	2003	10000	38	0

Tabla 5. Correctness by Construction Project Metrics Fuente: (Chapman, 2005).

14 Líneas de código
15 Real time air traffic information system at the London Terminal Control Centre
16 Ship/Helicopter operating limits information system developed to UK MoD
17 Certification Authority for Smart card operating system maintained by Mastercard
18 A UK military stores management system
19 NSA Tokeneer ID Station demonstrator biometrics system

CAPÍTULO III

DISEÑO E IMPLEMENTACIÓN

En el presente capítulo se detallan los pormenores que tienen los dos proyectos a estudiar del cual el primer proyecto ha sido elaborado sin utilizar normas o estándares de seguridad y el segundo proyecto ha sido elaborado implementando la norma ISO 27001 con soporte en la metodología DSL las cuales ofrecen aseguramiento a todo el proceso del ciclo de vida del proyecto.

3.1. Detalle del primer proyecto

3.1.1. Descripción

El sistema de integrar los datos provistos por un sistema contable e inventario denominado “BONES”, el cual mantiene los datos tanto de los equipos como de los clientes. Para la generación de una orden de compra se debe asignar la nomenclatura de ORDEN DE COMPRA provista por sistema rotativo de asignaciones de órdenes de compra (SERCOP, 2017) y los datos de la factura. El sistema una vez que haya agregado la orden de compra puede visualizar un detalle con los datos del cliente y los equipos respectivos, a esta orden se debe incluir un serial único para cada equipo. Como parte del sistema se debe calendarizar un cronograma de mantenimientos preventivos para cada producto. Además, el sistema debe generar documentos como los siguientes, cronograma de mantenimientos, garantía de vigencia, certificado de equipos con sus respectivos seriales.

Después de hacer el seguimiento a cada orden de compra se debe cumplir con el cronograma de mantenimientos, el cual genera una ORDEN DE SERVICIO, que no es más el mantenimiento preventivo que está sujeto a la ORDEN DE COMPRA, para que una garantía siga en vigencia. Como parte de la orden de servicio se adjuntan informes con el detalle individual del equipo verificado, y adicional a este informe se debe generar un informe general que es la compilación de todos los informes individuales.

3.1.2. Metodología utilizada

Por la cualidad del producto se eligió la metodología XP que se basa en el costo, el tiempo, calidad y alcance. Como XP mantiene un ciclo de vida dinámico se ha establecido trabajar con las siguientes fases, exploración, planificación, iteración y puesta en producción.

3.1.2.1. Exploración

En las reuniones con los interesados se ha estimado un tiempo de desarrollo base las cuales se detalla en la tabla 6

Fase	Tiempo horas
Planificación	27
Diseño	23
Desarrollo	280
Pruebas	18
Total	348

Tabla 6. Tiempo base de primer proyecto

Además, se acordó con los interesados que pueden variar los tiempos de acuerdo al análisis a profundidad que necesite cada requerimiento.

3.1.2.2. Planificación

Se ha realizado reuniones con los interesados en las cuales se ha detallado las historias de los usuarios con descripciones cortas de lo que el sistema debe hacer. Con estos datos se ha podido recopilar y establecer requerimientos con los que los interesados estuvieron de acuerdo, los cuales se muestran en la tabla 4 de requerimientos del primer proyecto.

Orden establecido para cumplir con los requerimientos	Descripción
R1	Gestión de datos sistema BONES
R2	Gestión de tareas fijas
R3	Gestión de seriales
R4	Gestión de informes
R5	Gestión de plantilla de documentos
R6	Gestión de usuarios
R7	Manejo de garantía de equipo

Orden establecido para cumplir con los requerimientos	Descripción
R8	Manejo de cronograma de mantenimiento
R9	Manejo de orden de compra
R10	Manejo de orden de servicio

Tabla 7. Requerimientos del primer proyecto

Con los requerimientos obtenidos se acordó en un plan de entregas el cual está determinado en el orden de la tabla 4, para cada requerimiento se ha establecido un plazo de una semana. Como parte esencial de la planificación se toma en cuenta la comunicación que se debe tener con los interesados, en este caso cliente y desarrolladores, la misma debe ser constante y clara, como lo afirman Ghani & Yasin (2013). Sin embargo, se acordó que las reuniones se realizaran dos veces a la semana una al inicio y otra a la mitad de la semana.

Como parte del análisis se ha establecido roles que el usuario puede tener durante su actividad dentro de la aplicación, los privilegios asignados donde L es lectura, C es creación, M es modificación y E es eliminación, en la tabla 8 se detallan los roles y privilegios asignados.

Roles / Privilegios	Administrador				Técnico				Ventas			
	L	C	M	E	L	C	M	E	L	C	M	E
Gestión de datos sistema BONS	x	x	x	x								
Gestión de tareas fijas	x	x	x		x	x	x	x				
Gestión de seriales	x	x	x	x					x	x	x	
Gestión de informes	x				x							
Gestión de plantilla de documentos	x	x	x	x	x		x		x		x	
Gestión de usuarios	x	x	x	x								
Manejo de garantía de equipo	x	x	x		x		x		x	x	x	x
Manejo de cronograma de mantenimiento	x	x	x		x				x			
Manejo de orden de compra	x	x	x	x					x	x	x	x
Manejo de orden de servicio	x	x	x	x	x	x	x	x	x			

Tabla 8. Roles y privilegios

3.1.2.3. Iteración

3.1.2.3.1. Arquitectura

Para este proyecto se ha especificado que debe tener una arquitectura de cliente servidor tal y como se muestra en la figura 10, cabe mencionar que el sistema trabaja con dos bases de datos; la primera base de datos sirve de consulta de datos de la factura, datos de cliente y datos de equipo. La segunda base interactúa con la aplicación para almacenamiento e interacción con sus respectivas gestiones de administración.

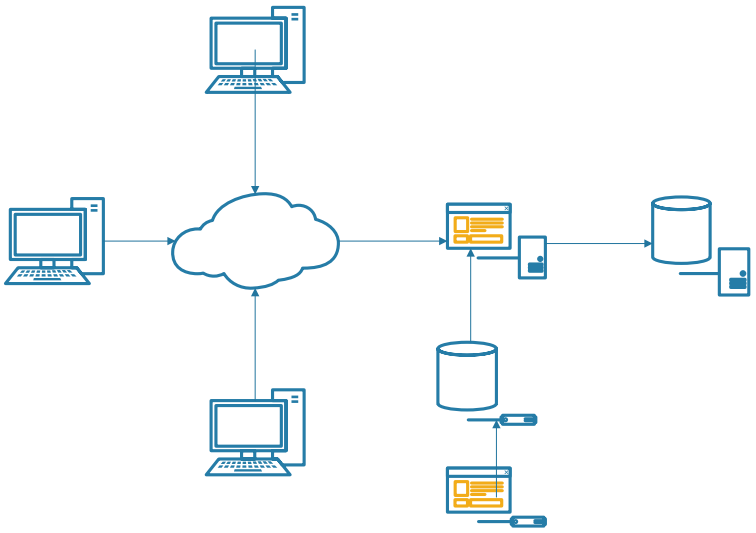

Figura 10. Arquitectura del primer proyecto

3.1.2.3.2. Riesgos

Los riesgos que se encontraron para realizar este proyecto se han analizado con el grupo de desarrollo de acuerdo a los requerimientos especificados por los interesados, como muestra la tabla 9.

Requerimiento	Riesgo
Gestión de datos sistema BONES	Parametrización de tipos de datos utilizados en interacción de base de datos
Gestión de tareas fijas	
Gestión de seriales	Generación única para equipos de acuerdo al modelo y tipo de producto

Requerimiento	Riesgo
Gestión de informes	
Gestión de plantilla de documentos	
Gestión de usuarios	
Manejo de garantía de equipo	
Manejo de cronograma de mantenimiento	Calendarización de acuerdo a tipo de producto sea este computador o impresora
Manejo de orden de compra	Vigencia de garantía de acuerdo a la fecha de compra
Manejo de orden de servicio	

Tabla 9. Riesgos de acuerdo a requerimientos

3.1.2.3.3. Diseño

Para el diseño se ha utilizado diagramas de clases para detallar la estructura del sistema mostrando sus atributos y métodos con los cuales va a trabajar, como muestra la figura 11.

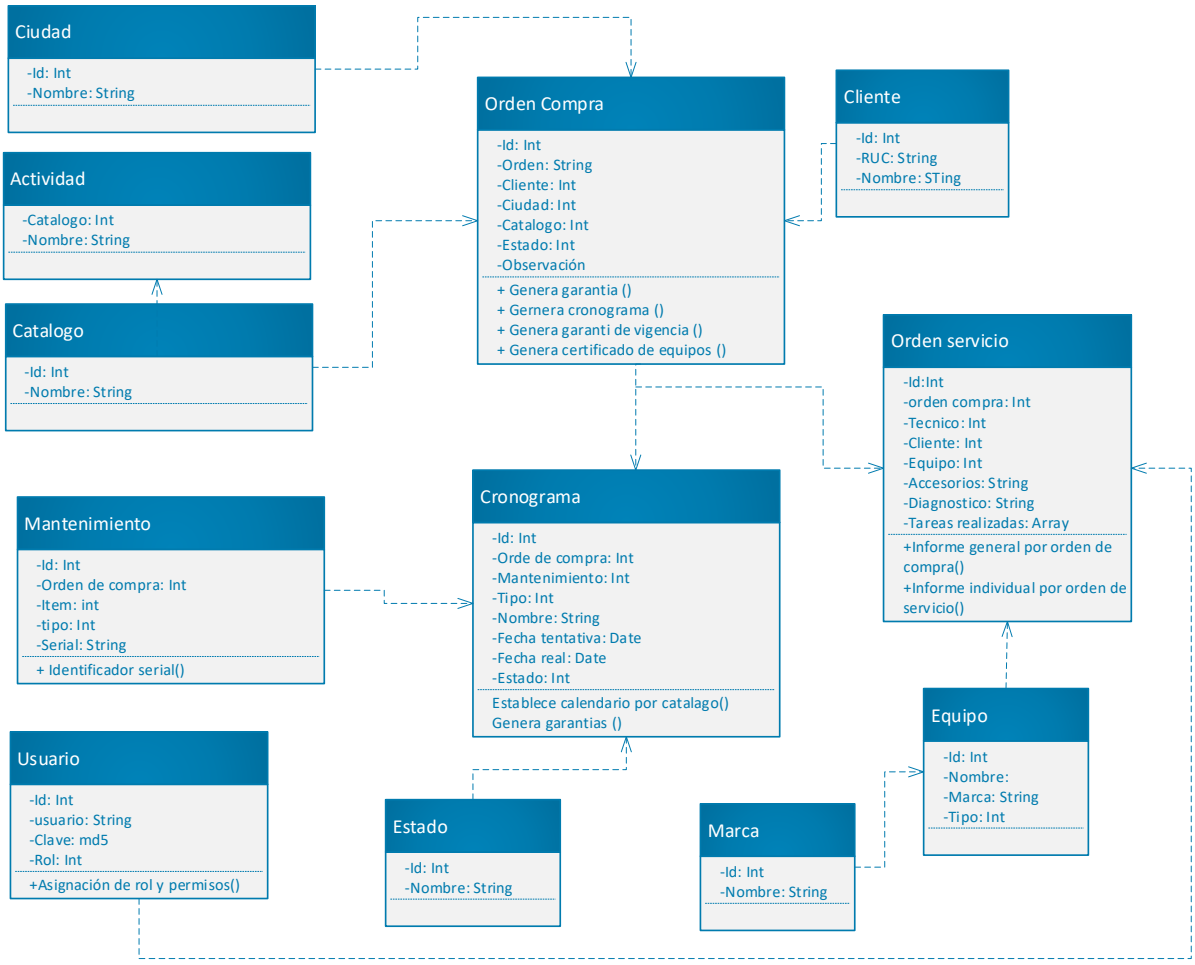

Figura 11. Diagrama de clases primer proyecto

Para tener una visibilidad clara de cómo funciona el sistema se ha desarrollado un diagrama de secuencia generalizado, donde se establece la interacción en conjunto de los objetos a través del tiempo, como muestra la figura 12.

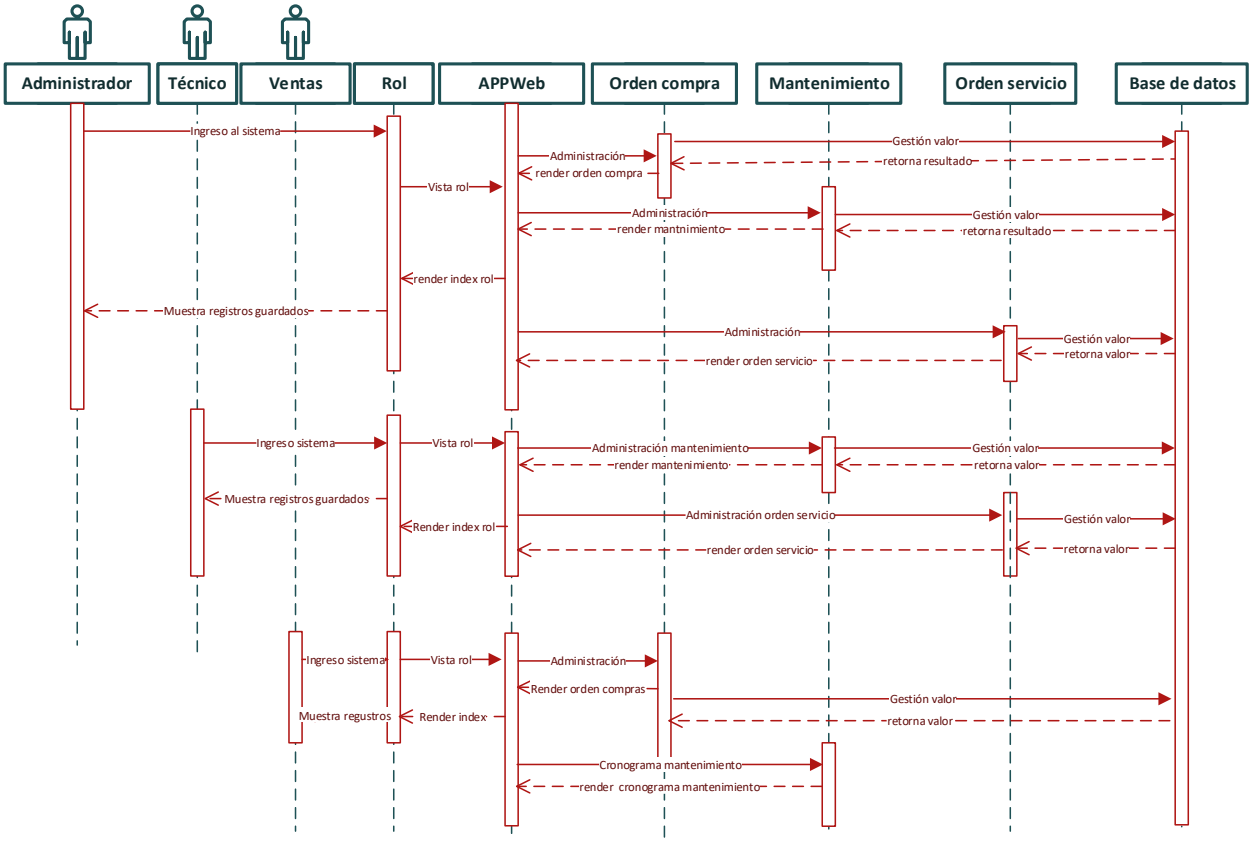

Figura 12. Diagrama de secuencia primer proyecto

3.1.2.3.4. Desarrollo de código

Para el desarrollo del código se ha establecido herramientas que ayuden al desarrollo de la codificación, como muestra la tabla 10.

Tipo de herramienta	Nombre	Versión
IDE	Notepad	7.5.9
Framework de desarrollo	Yii2	2.0.4
Framework web (estilos)	Bootstrap	3.1
Manejador de dependencias	Composer	2.0.1
Cliente SFTP	WinSCP	5.09

Tabla 10. Herramientas de codificación

Para la codificación se ha establecido que se trabaja 30 horas por semana, donde el equipo de desarrollo no se traslada al lugar donde se encuentran los interesados, el equipo de

desarrollo se divide en dos grupos integrados de dos personas cada uno, y se ha utilizado los estándares de programación.

3.1.2.4. Puesta en producción

Existe un riesgo para este proyecto ya que no se hicieron las pruebas respectivas en un ambiente de desarrollo, sino que las pruebas se las realizaron en ambiente producción directamente, la detección de errores y la corrección de estos se han elaborado mientras utilizaban la aplicación. Las pruebas de aceptación o funcionales se las realizaron en base a las historias de los usuarios, donde se han tomado en cuenta diversos escenarios para comprobar la historia del usuario. Donde el interesado verifica que los resultados de estas pruebas sean correctos y de la misma manera especificar el orden de prioridad los errores sean corregidos

3.2. Detalle del segundo proyecto

3.2.1. Descripción

El planificador como se le conoce al sistema tiene como objetivo brindar al docente una plantilla de planificación de unidad y otra de planificación anual de currículo de acuerdo a la asignatura y grado que el docente sea asignado en su institución educativa.

Una planificación curricular debe ayudar al docente a realizar sus actividades durante el año lectivo Ministerio de Educación (2018), por esta razón los interesados en proporcionar esta herramienta a los docentes, establecen que debe haber una guía en la que los docentes se basen para hacer sus propias planificaciones o adoptar las planificaciones que son elaboradas por parte del interesado.

Cabe recalcar que existen dos tipos de planificaciones, la planificación de unidad didáctica que realiza un análisis de los conceptos, criterios, y herramientas con las cuales enseñará y evaluará al estudiante durante cada unidad de su planificación; la planificación curricular anual es la compilación de las planificaciones unitarias de acuerdo a la asignatura y al grado.

3.2.2. Metodología adoptada

La metodología adoptada para el desarrollo del sistema se ha establecido en XP, ya que para el equipo de desarrollo tiene experiencia en esta metodología, y por consecuencia sabe que se debe tener una comunicación muy estrecha con los interesados para proveer instrucciones a detalle. Además de la metodología XP se ha adoptado la metodología DSL la cual establece seguridad en el ciclo de vida del sistema, y también trabaja mediante fases en el desarrollo del sistema como muestra la figura 13.

Figura 13. Ciclo de vida SDL Fuente: Microsoft Corporation

3.2.3. Formación

Para la formación en el aspecto de seguridad se ha utilizado la implementación de la norma 27002:2013, específicamente con la ayuda del Anexo A de la norma 27001:2013, la cual provee una guía en para el control de los dominios, en la tabla 11 se establece un resumen de actividades realizadas con la ayuda de los dominios de control

Dominios de control		Actividad
A5	Políticas de seguridad de la información	
A5.1	Directrices de gestión de la seguridad de la información	
A5.1.1	Políticas para la seguridad de la información	La empresa cuenta con una política de seguridad donde se establece los riesgos de información y el control de áreas relevantes, también se especifica la cobertura en activos sean estos de información, sistemas o servicios
A5.1.2	Revisión de las políticas para la seguridad de la información	Las políticas son revisadas cada año por los directores de cada área para la actualización y mejora en el nivel de madurez

Dominios de control		Actividad
A6	Organización de la seguridad de la información	
A6.1	Organización interna	
A6.2	Los dispositivos móviles y el teletrabajo	
A6.2.1	Política de dispositivos móviles	Se han establecido políticas para el uso de dispositivos móviles, donde se controlan los sistemas portátiles estos a su vez garantizan la actualización y soporte al dispositivo
A6.2.2	Teletrabajo	Se ha integrado una virtualización para el acceso a servidores donde cuenta una autenticación 2FA
A7	Seguridad relativa a los recursos humanos	
A7.1	Antes del empleo	
A7.1.1	Investigación de antecedentes	La empresa cuenta con el asesoramiento para la contratación d personal donde cumplen altos estándares
A7.1.2	Términos y condiciones del empleo	Las condiciones de empleo se dictaminan claramente para la plaza de trabajo que se necesite
A7.2	Durante el empleo	
A7.2.1	Responsabilidades de gestión	Las responsabilidades están contempladas en las condiciones del trabajo adicionalmente se establece educación en seguridad, esta se basa en el conocimiento adoptado previamente en la empresa
A7.2.2	Concienciación, educación y capacitación en seguridad de la información	Existe manuales de concienciación que permiten capacitarse sobre seguridad a todos los trabajadores
A7.2.3	Proceso disciplinario	No cuenta con un proceso disciplinario para su ejecución si alguno de los miembros faltare a su cumplimiento
A7.3	Finalización del empleo o cambio en el puesto de trabajo	

Dominios de control		Actividad
A7.3.1	Responsabilidades ante la finalización o cambio	Se establece que cada responsable debe entregar toda información de seguridad necesaria para que el proceso de seguridad se mantenga
A8	Gestión de activos	
A8.1	Responsabilidad sobre los activos	
A8.1.1	Inventario de activos	Existe un inventario de activos donde cuenta con datos digitales, software, infraestructura, servicios de información y proveedores.
A8.1.2	Propiedad de los activos	como parte del inventario establece propietarios en caso de riesgo y el reporte de incidentes de seguridad
A8.1.3	Uso aceptable de los activos	No cuenta con un proceso que describa el uso aceptable de los recursos tecnológicos
A8.1.4	Devolución de activos	No cuenta con una guía que mencione un procedimiento para la recuperación de los activos tras baja o despido
A8.2	Clasificación de la información	
A8.2.1	Clasificación de la información	La clasificación de la información esta provista por un manual donde se basa los requisitos de confidencialidad, integridad y responsabilidad
A8.2.2	Etiquetado de la información	La información conserva una etiquetación de acuerdo a la clasificación de la información
A8.2.3	Manipulado de la información	La información es manipulada de acuerdo a la clasificación y depende a que personal este asignado
A8.3	Manipulación de los soportes	
A8.3.1	Gestión de soportes extraíbles	Existe un manual donde se establece el registro de medio extraíbles
A8.3.2	Eliminación de soportes	Se establece que cada año se debe dar de baja a soportes extraíbles
A8.3.3	Soportes físicos en tránsito	No existe medidas en las cuales se puede tener una respuesta ante un incidente de tránsito de soporte físico
A9	Control de acceso	

Dominios de control		Actividad
A9.1	Requisitos de negocio para el control de acceso	
A9.1.1	Política de control de acceso	Cuenta con una política de control de acceso
A9.1.2	Acceso a las redes y a los servicios de red	Existe segregación de funciones a servicios de red
A9.2	Gestión de acceso de usuario	
A9.2.1	Registro y baja de usuario	Todos los usuarios tienen registro y se mantiene control en cada uno de ellos
A9.2.2	Provisión de acceso de usuario	Existe responsables en asignar permisos a los usuarios
A9.2.3	Gestión de privilegios de acceso	De acuerdo a los roles que ejerce mantiene su perfil
A9.2.4	Gestión de la información secreta de autenticación de los usuarios	No cuenta con controles técnicos
A9.2.5	Revisión de los derechos de acceso de usuario	Existe una revisión periódica de los derechos de acceso a cada usuario
A9.2.6	Retirada o reasignación de los derechos de acceso	El manual cuenta el procedimiento para la retirada o modificación de acceso
A9.3	Responsabilidades del usuario	
A9.3.1	Uso de la información secreta de autenticación	Existe procesos para perdidas de información confidencial
A9.4	Control de acceso a sistemas y aplicaciones	
A9.4.1	Restricción del acceso a la información	Existen controles de acceso para cada usuario
A9.4.2	Procedimientos seguros de inicio de sesión	Existe un procedimiento que en la cual establece el control del inicio de sesión
A9.4.3	Sistema de gestión de contraseñas	Existe una política donde se establece la criticidad en la gestión de contraseñas
A9.4.4	Uso de utilidades con privilegios del sistema	Los usuarios son responsables de cada privilegio que ha sido asignado
A9.4.5	Control de acceso al código fuente de los programas	Se ha creado un manual para el manejo de código fuente y como se puede modificar el código
A10	Criptografía	
A10.1	Controles criptográficos	

Dominios de control		Actividad
A10.1.1	Política de uso de los controles criptográficos	No cuenta con una política, sin embargo, se toman niveles de cumplimiento para el uso de control criptográfico
A10.1.2	Gestión de claves	Las llaves tanto públicas como privadas se manejan de acuerdo a la gestión de claves
A11	Seguridad física y del entorno	
A11.1	Áreas seguras	
A11.1.1	Perímetro de seguridad física	Las instalaciones cuentan con un espectro de seguridad física adecuada para la jerarquía de la empresa
A11.1.2	Controles físicos de entrada	Cuentan con bajos controles físicos en entrada
A11.1.3	Seguridad de oficinas, despachos y recursos	Dependiendo a los activos es establecido la seguridad de la oficina
A11.1.4	Protección contra las amenazas externas y ambientales	No cuenta con una política de protección contra amenazas externa ni ambientales
A11.1.5	El trabajo en áreas seguras	No cuenta con una política que abarque la verificación de carácter sensible
A11.1.6	Áreas de carga y descarga	No cuenta con áreas de carga y descarga
A11.2	Seguridad de los equipos	
A11.2.1	Emplazamiento y protección de equipos	Los equipos se encuentran en áreas protegidas, donde se establecen controles para mitigar riesgos y amenazas físicas
A11.2.2	Instalaciones de suministro	Existen suministros de electricidad en caso de fallo de la fuente principal. Su plan establece el manejo y control de este dispositivo
A11.2.3	Seguridad del cableado	
A11.2.4	Mantenimiento de los equipos	Las instalaciones cuentan con el mantenimiento necesario
A11.2.5	Retirada de materiales propiedad de la empresa	No existe relativos al traslado de activos de información
A11.2.6	Seguridad de los equipos fuera de las instalaciones	No existe políticas que establezca el uso fuera de la instalación sin embargo cuenta con aseguramiento en todos los dispositivos
A11.2.7	Reutilización o eliminación segura de equipos	No existe una política o proceso que cubra esta reutilización
A11.2.8	Equipo de usuario desatendido	No aplica

Dominios de control		Actividad
A11.2.9	Política de puesto de trabajo despejado y pantalla limpia	No se establece como una política
A12	Seguridad de las operaciones	
A12.1	Procedimientos y responsabilidades operacionales	
A12.1.1	Documentación de procedimientos operacionales	Política para procedimientos de operación donde cuenta versión, capacidad, rendimiento incidentes copias de seguridad, restauración
A12.1.2	Gestión de cambios	Política con un control de cambios donde se evalúan riesgos potenciales
A12.1.3	Gestión de capacidades	Política donde se especifica la capacidad donde se cuenta con seguimiento de métricas relevantes y la respectiva notificación en casos críticos
A12.1.4	Separación de los recursos de desarrollo, prueba y operación	No aplica
A12.2	Protección contra el software malicioso (malware)	
A12.2.1	Controles contra el código malicioso	Políticas donde se tienen en cuenta todos los activos y la manera de actualización
A12.3	Copias de seguridad	
A12.3.1	Copias de seguridad de la información	Se genera copia de seguridad de manera incremental cada semana
A12.4	Registros y supervisión	
A12.4.1	Registro de eventos	Procedimiento donde incluye permisos de usuario, actividades realizadas, identidad de lugar y dispositivo
A12.4.2	Protección de la información del registro	no aplica
A12.4.3	Registros de administración y operación	no aplica
A12.4.4	Sincronización del reloj	Existe control en sincronización de reloj tomando en cuenta el horario de Guayaquil como referencia

Dominios de control		Actividad
A12.5	Control del software en explotación	
A12.5.1	Instalación del software en explotación	No aplica
A12.6	Gestión de la vulnerabilidad técnica	
A12.6.1	Gestión de las vulnerabilidades técnicas	No aplica
A12.6.2	Restricción en la instalación de software	No aplica
A12.7	Consideraciones sobre la auditoria de sistemas de información	
A12.7.1	Controles de auditoría de sistemas de información	
A13	Seguridad de las comunicaciones	
A13.1	Gestión de la seguridad de las redes	
A13.1.1	Controles de red	Existen una política donde se especifica administración monitoreo y cumplimento de la misma
A13.1.2	Seguridad de los servicios de red	Las redes están clasificadas y existe monitoreo frecuente
A13.1.3	Segregación en redes	Cuenta con segregación de redes
A13.2	Intercambio de información	
A13.2.1	Políticas y procedimientos de intercambio de información	Procedimiento que contempla correo y acceso ftp
A13.2.2	Acuerdos de intercambio de información	No aplica
A13.2.3	Mensajería electrónica	Existe política de seguridad donde se establece el intercambio de datos de comunicación de red
A13.2.4	Acuerdos de confidencialidad o no revelación	Existe una política que hace referencia al uso y de la información con respecto a la divulgación basado en aspectos legales

Dominios de control		Actividad
A14	Adquisición, desarrollo y mantenimiento de los sistemas de información	
A14.1	Requisitos de seguridad en los sistemas de información	
A14.1.1	Análisis de requisitos y especificaciones de seguridad de la información	Se establece procedimientos para el análisis de requisitos, riesgos y arquitectura
A14.1.2	Asegurar los servicios de aplicaciones en redes públicas	Se ha establecido un procedimiento para el control de las aplicaciones en redes públicas
A14.1.3	Protección de las transacciones de servicios de aplicaciones	Se protege la información en protocolos seguros y cifrados
A14.2	Seguridad en el desarrollo y en los procesos de soporte	
A14.2.1	Política de desarrollo seguro	Se establece una política para el desarrollo seguro
A14.2.2	Procedimiento de control de cambios en sistemas	procedimiento donde incluye planificación y prueba de cambios donde s incluye cambios de emergencia
A14.2.3	Revisión técnica de las aplicaciones tras efectuar cambios en el sistema operativo	Se han evaluado instancias de riesgos y su respectiva solución
A14.2.4	Restricciones a los cambios en los paquetes de software	Se mantiene un inventario con los activos y su control para actualización y dada de baja
A14.2.5	Principios de ingeniería de sistemas seguros	Se han establecido una metodología de desarrollo seguro
A14.2.6	Entorno de desarrollo seguro	Se ha establecido una política para el manejo del software
A14.2.7	Externalización del desarrollo de software	Se mantiene un procedimiento para la propiedad del código y sus respectivos controles de seguridad
A14.2.8	Pruebas funcionales de seguridad de sistemas	Las pruebas funcionales se hacen de acuerdo a los requerimientos presentados
A14.2.9	Pruebas de aceptación de sistemas	Se genera pruebas de aceptación de acuerdo a los requerimientos presentados

Dominios de control		Actividad
A14.3	Datos de prueba	
A14.3.1	Protección de los datos de prueba	No aplica
A15	Relación con proveedores	
A15.1	Seguridad en las relaciones con proveedores	
A15.1.1	Política de seguridad de la información en las relaciones con los proveedores	Existe un manual para el trato de información de datos con proveedores
A15.1.2	Requisitos de seguridad en contratos con terceros	Los acuerdos mantienen la gestión de relaciones y riesgos y se establece una descripción de la información
A15.1.3	Cadena de suministro de tecnología de la información y de las comunicaciones	No aplica
A15.2	Gestión de la provisión de servicios del proveedor	
A15.2.1	Control y revisión de la provisión de servicios del proveedor	No aplica
A15.2.2	Gestión de cambios en la provisión del servicio del proveedor	No aplica
A16	Gestión de incidentes de seguridad de la información	
A16.1	Gestión de incidentes de seguridad de la información y mejoras	
A16.1.1	Responsabilidades y procedimientos	Existe procedimiento para la gestión de incidentes
A16.1.2	Notificación de los eventos de seguridad de la información	La notificación se informa eventualmente de acuerdo a la magnitud
A16.1.3	Notificación de puntos débiles de la seguridad	No se aplica

Dominios de control		Actividad
A16.1.4	Evaluación y decisión sobre los eventos de seguridad de información	Se ha clasificado los incidentes para evaluar eventos
A16.1.5	Respuesta a incidentes de seguridad de la información	De acuerdo a la clasificación de incidentes se mantiene respuesta
A16.1.6	Aprendizaje de los incidentes de seguridad de la información	Se actualiza los incidentes prevenidos para mejorar la seguridad
A16.1.7	Recopilación de evidencias	Se mantienen recopilaciones de los riesgos
A17	Aspectos de seguridad de la información para la gestión de la continuidad de negocio	
A17.1	Continuidad de la seguridad de la información	
A17.1.1	Planificación de la continuidad de la seguridad de la información	Existe un manual de continuidad dependiendo al impacto potencial de los incidentes
A17.1.2	Implementar la continuidad de la seguridad de la información	Existen planes con plazos definidos para restauración de servicios
A17.1.3	Verificación, revisión y evaluación de la continuidad de la seguridad de la información	Se establece un método de pruebas y la frecuencia
A17.2	Redundancias	
A17.2.1	Disponibilidad de los recursos de tratamiento de la información	Cuenta con disponibilidad recursiva para el manejo de la información
A18	Cumplimiento	
A18.1	Cumplimiento de los requisitos legales y contractuales	
A18.1.1	Identificación de la legislación aplicable y de los requisitos contractuales	Cuenta con una política que da cumplimiento en aspectos y requisitos legales
A18.1.2	Derechos de Propiedad Intelectual (DPI)	No se establece derecho de propiedad intelectual

Dominios de control		Actividad
A18.1.3	Protección de los registros de la organización	Existe una política que contemple la protección del registro de la empresa
A18.1.4	Protección y privacidad de la información de carácter personal	No cuenta con regulación
A18.1.5	Regulación de los controles criptográficos	No cuenta con regulación
A18.2	Revisiones de la seguridad de la información	
A18.2.1	Revisión independiente de la seguridad de la información	No existe implementación de controles aliadas a los riesgos de información
A18.2.2	Cumplimiento de las políticas y normas de seguridad	Se establece un proceso de cumplimiento y verificación periódica
A18.2.3	Comprobación del cumplimiento técnico	Se establece un proceso de cumplimiento y verificación técnica

Tabla 11. Aplicación de dominios de control a segundo proyecto

3.2.4. Requisitos

El equipo de desarrollo ha realizado reuniones con los interesados cuyos perfiles son, docentes, creadores de libros, planificadores de currículo, y personal administrativo, en las reuniones se han tomado en cuenta las historias de los usuarios con descripciones cortas de lo que el sistema debe hacer. Con el análisis de las historias se pudo recopilar los siguientes requerimientos, los mismos que son aprobados por los interesados, al igual que el anterior proyecto la tabla 12 detalla por columna el orden y el requerimiento.

Orden de entrega del requerimiento	Requerimiento
R1	Gestión de nivel académico
R2	Gestión de subnivel académico
R3	Gestión de área curricular
R4	Gestión de asignatura
R5	Gestión de objetivos de área
R6	Gestión de objetivos de grado
R7	Gestión de criterios de evaluación
R8	Gestión de destrezas

Orden de entrega del requerimiento	Requerimiento
R9	Gestión de indicadores
R10	Gestión de grado
R11	Gestión de colección
R12	Gestión de libro
R13	Gestión de Unidad
R14	Gestión de lección
R15	Gestión de plantillas de documentos
R16	Gestión de Mensajes a presentar
R17	Gestión de usuarios
R18	Manejo y control de planificación de unidad didáctica
R19	Manejo y control de plan curricular anual

Tabla 12. Requerimientos del segundo proyecto

3.2.4.1. Arquitectura de la aplicación

La arquitectura utilizada en este proyecto es cliente - servidor, con ayuda de una función REST API para la creación y asignación de asignatura y grado para el docente, la conexión entre sistemas establece un gran sistema para el manejo de las diferentes aplicaciones que tiene, como muestra la figura 14 se establece pequeños sistemas para la visualización en un solo ambiente web.

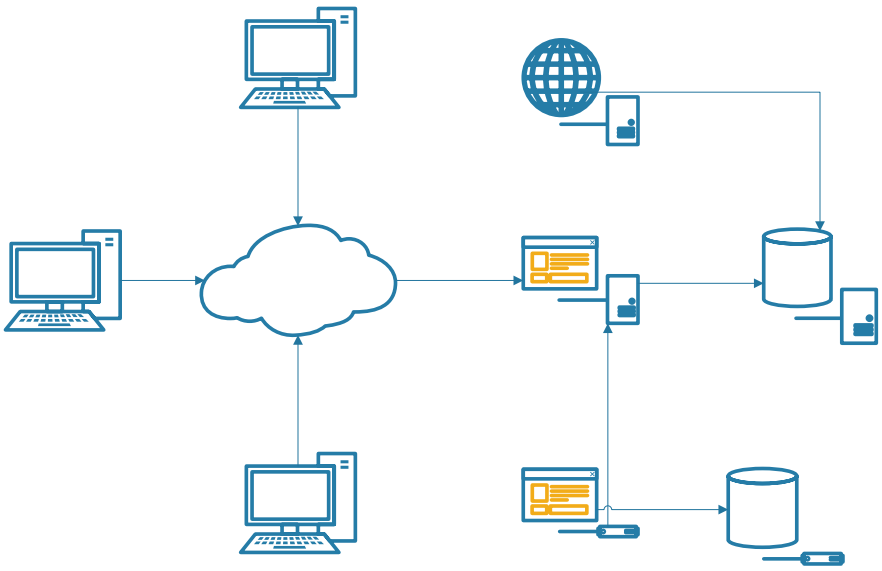

Figura 14. Arquitectura segundo proyecto

3.2.4.2. Plataforma

Para el segundo proyecto por parte de los interesados se ha escogido un servidor de Amazon AWS en el cual está instalado los siguientes aspectos como detalla la tabla 13

Detalle	Nombre	Versión
Servidor	Linux	7.1.30
Servidor HTTP	Apache	2.5
Servidor base de datos	Mysql /mariaBD	5.7.25 / 10.1
Lenguaje de programación	PHP	7.2.20 y 7.3.7

Tabla 13. Detalle de la plataforma

3.2.4.3. Tipos de datos

Los datos que se muestran en la aplicación web son confidenciales cuando se construye el formato de impresión y son públicos cuando el docente descarga o interactúa con ese formato.

3.2.4.4. Requerimientos con marcos regulatorios

Como se presenta en la fase de entrenamiento se ha aplicado la norma ISO 27002:2013 con la metodología de desarrollo seguro en el ciclo de vida (DSL).

3.2.4.5. Tipos de registro

Acceso a recursos diseñados para cada perfil y el uso de los privilegios para los cuales tiene permitido cada rol.

3.2.4.6. Perfiles de usuario

Una vez desarrollado los análisis de requerimientos se ha encontrado la utilización de los siguientes roles: Administrador, planificador y docente

3.2.4.7. Tipos de acceso a los datos por perfil

Se ha definido los accesos a cada rol como muestra la tabla 14 donde, se establece los parámetros de acceso a los datos y de acuerdo a su nomenclatura L tiene acceso de lectura, C acceso para crear, M acceso para editar, E acceso a eliminación de datos

Roles/Privilegios	**Administrador**				**Planificador**				**Docente**			
	L	**C**	**M**	**E**	**L**	**C**	**M**	**E**	**L**	**C**	**M**	**E**
Gestión de nivel académico	x	x	x	x								
Gestión de subnivel académico	x	x	x	x								
Gestión de área curricular	x	x	x	x								
Gestión de asignatura	x		x	x								
Gestión de objetivos de área	x	x	x	x	x		x					
Gestión de objetivos de grado	x	x	x	x	x		x					
Gestión de criterios de evaluación	x	x	x	x	x		x					
Gestión de destrezas	x	x	x	x	x		x					
Gestión de indicadores	x	x	x	x	x		x					
Gestión de grado	x	x	x	x								
Gestión de colección	x	x	x	x								
Gestión de libro	x	x	x	x								
Gestión de Unidad	x	x	x	x								
Gestión de lección	x	x	x	x								
Gestión de plantillas de documentos	x											
Gestión de Mensajes a presentar	x	x	x									
Gestión de usuarios	x	x	x	x								
Manejo y control de planificación de unidad didáctica	x	x	x		x	x	x	x	x	x	x	
Manejo y control de plan curricular anual	x	x	x		x	x	x	x	x	x	x	

Tabla 14. Acceso de datos por perfil

3.2.4.8. Modo de autenticación

La autenticación se la realiza mediante usuario que es su correo personal o institucional conjuntamente con un password provisto para cada usuario, además cuanta con una asignación de tokens de acceso y recuperación de contraseña. El sistema cuenta con doble factor de seguridad, sin embargo, como el sistema interactúa por medio de otro sistema la autenticación se la realiza mediante los datos provistos por el API que brinda el sistema que los interconecta

3.2.5. Diseño

Para el diseño se ha utilizado diagramas de clases para detallar la estructura del sistema mostrando sus atributos y métodos con los cuales va a trabajar, como muestra la figura 15.

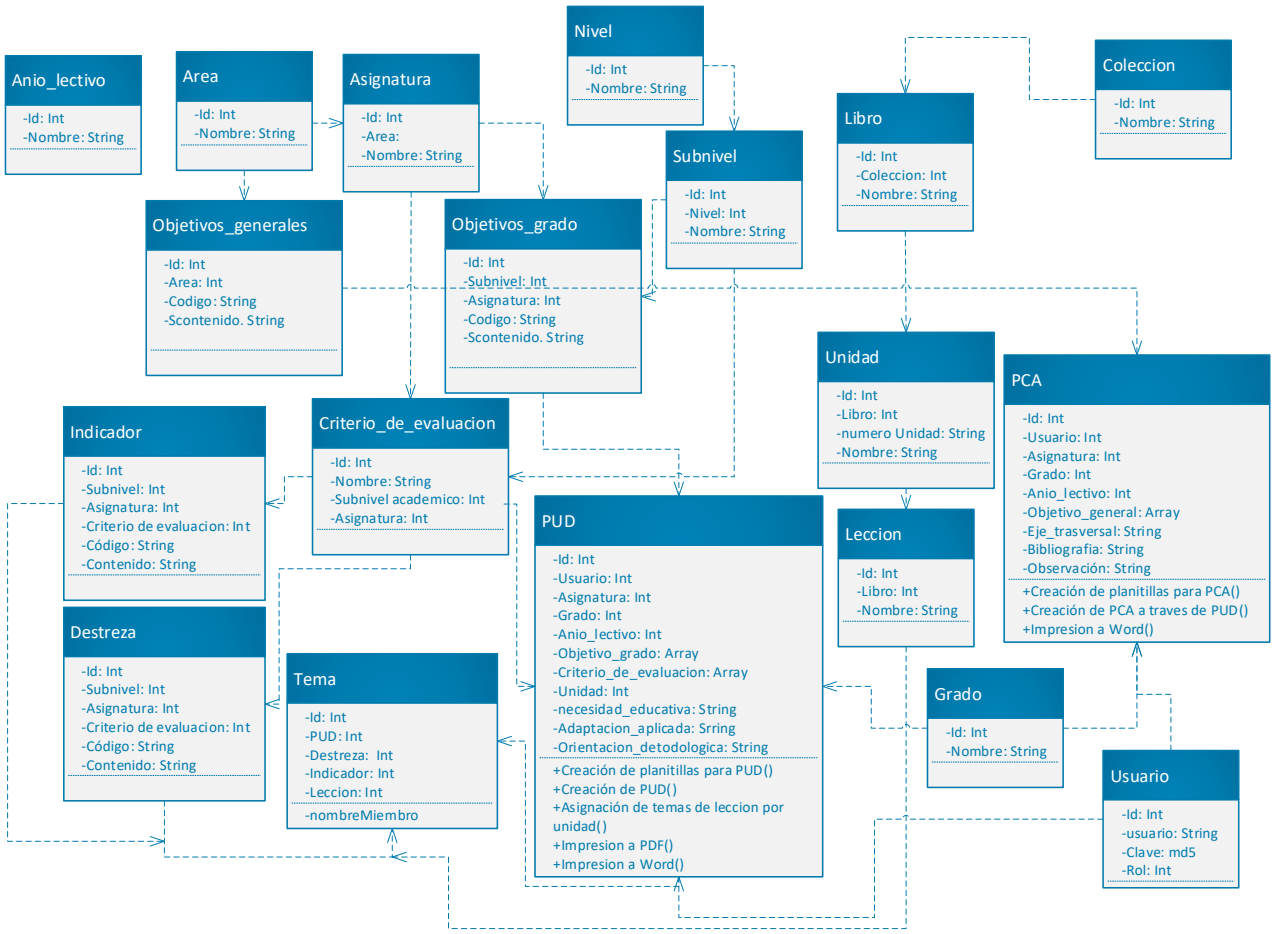

Figura 15. Diagrama de clases segundo proyecto

Para tener una visión más clara de cómo funciona el sistema se ha elaborado de forma general un diagrama de secuencia, donde se establece la interacción en conjunto de los objetos a través del tiempo, como muestra la figura 16.

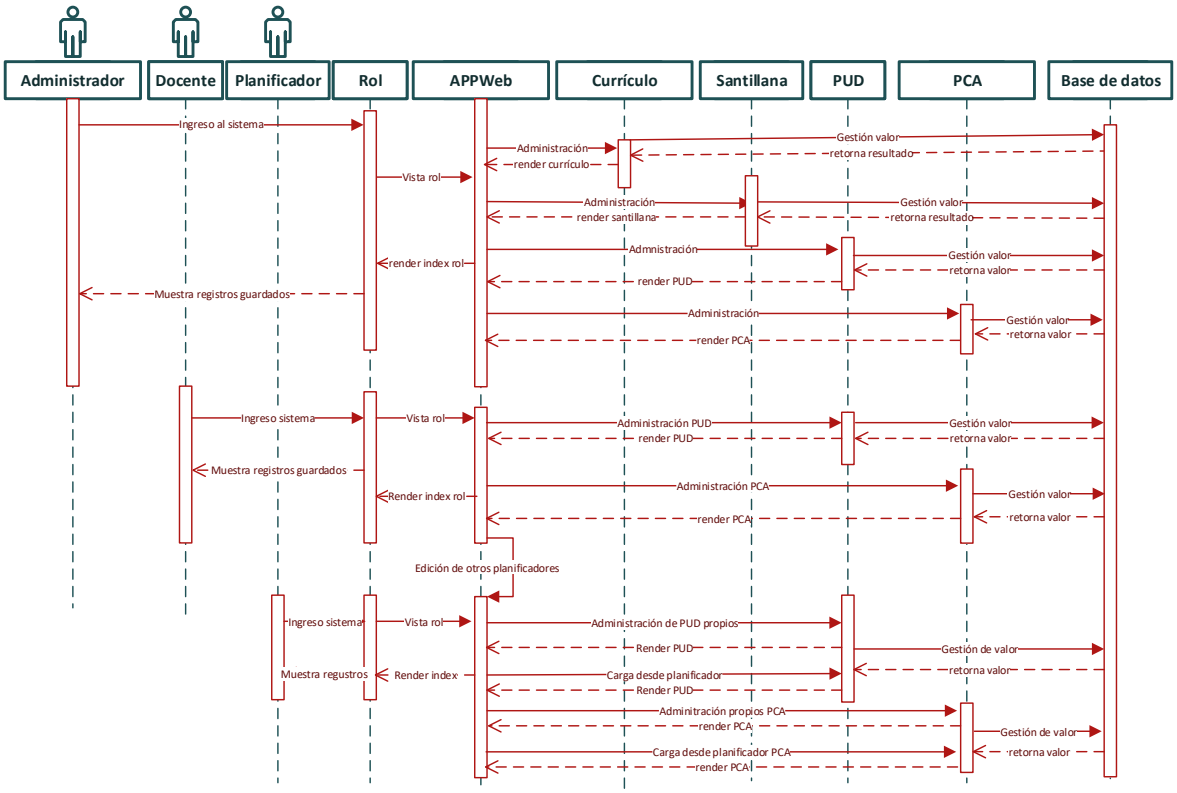

Figura 16. Diagrama de secuencia segundo proyecto

3.2.5.1. Análisis de riesgo

De acuerdo a las prácticas recomendadas por parte de la metodología DSL se establece que se debe realizar un análisis de reducción de superficies de ataque como muestra la siguiente tabla 15.

Componentes	Valor de amenaza	Respuesta
Bloquear contenido ejecutable del cliente de correo electrónico y el correo web	Alto	Revisar y actualizar las configuraciones apropiadas de acuerdo a las advertencias de seguridad y siga un proceso de gestión de parches Use una plataforma minimalista sin funcionalidades Aplicar una arquitectura segmentada que proporcione una separación efectiva

Componentes	Valor de amenaza	Respuesta
Bloquear la creación de procesos secundarios en todas las consultas	Alto	Utilizar frameworks seguros que, por diseño, automáticamente codifican el contenido para prevenir XSS
Bloquear la creación de contenido ejecutable en contenido http	Alto	Codificar los datos de requerimientos HTTP no confiables en los campos de salida HTML Aplicar codificación sensitiva al contexto, cuando se modifica el documento en el navegador del cliente, ayuda a prevenir DOM XSS
Bloquear la inyección de código en otros procesos	Alto	Realizar validaciones de entradas de datos en el servidor, utilizando listas blancas Realizar un escape caracteres especiales utilizando la sintaxis de caracteres específica para el intérprete
Bloquear JavaScript para que no inicien contenido descargado ejecutable	Medio	Utilizar formatos de datos menos complejos como JSON y evite la serialización de datos confidenciales. Actualizarlos procesadores y bibliotecas XML que utilice la aplicación o el sistema subyacente
Bloquear la ejecución de scripts potencialmente cifrados	Alto	Implementar verificaciones de integridad tales como firmas digitales en cualquier objeto serializado, con el fin de detectar modificaciones no autorizadas. Registrar las excepciones y fallas en la deserialización.
Bloquear la ejecución de los archivos ejecutables a menos que cumplan con un criterio de predominio, edad o lista de confianza	Medio	Remover dependencias, funcionalidades, componentes, archivos y documentación innecesaria y no utilizada Obtener componentes únicamente de orígenes oficiales utilizando canales seguros
Usar protección avanzada contra ransomware	Medio	Establecer una monitorización y alerta efectivos de tal manera que las actividades sospechosas sean detectadas y respondidas dentro de períodos de tiempo aceptables. Asegurar que todas las transacciones de alto valor poseen una traza de auditoría con

Componentes	Valor de amenaza	Respuesta
Bloquear el robo de credenciales del sistema	Alto	controles de integridad que permitan detectar su modificación o borrado. Implementar autenticación multi-factor para evitar ataques automatizados No utilice credenciales por defecto en su software. Implementar controles contra contraseñas débiles. Limitar o incrementar el tiempo de respuesta de cada intento fallido de inicio de sesión. Registrar todos los fallos y avise a los administradores cuando se detecten ataques de fuerza bruta.
Bloquear la creación de procesos secundarios en el envío de consultas	Medio	Clasificar los datos procesados, almacenados o transmitidos por el sistema. Cifrar todos los datos sensibles cuando sean almacenados Deshabilitar el almacenamiento en cache de datos sensibles
Bloquear la persistencia mediante la suscripción de eventos	Bajo	Deshabilitar el listado de directorios del servidor web y asegúrese que los metadatos/fuentes de archivos. Registrar y alertar errores de control de acceso. Limitar tasa de acceso a las APIs para minimizar el daño de herramientas de ataque automatizadas

Tabla 15. Análisis de riesgos

Para la identificación de riesgos y amenazas en el software se requiere un esquema estructurado y repetible según Castellaro, Romaniz, Ramos y Pessolani (2016), el modelado de amenazas constituye un marco de trabajo para el proceso de análisis de riesgo estructurado que identifica las amenazas de una aplicación, y en tal virtud dar valor a los riesgos que está expuesta. Para este modelado ha trabajado el equipo de desarrollo incluido los jefes de proyecto y los involucrados de la empresa contratante donde se han catalogado los riesgos que pueden suscitar y se identificaron los componentes clave de amenazas asignando un valor cuantitativo de riesgo, además se identificó como responder a las amenazas.

Durante este el análisis de requerimientos se enfatizó en los requerimientos funcionales y de seguridad, poniendo énfasis en los primeros, dejando de lado otro tipo de requerimientos.

Como aspecto de respuesta ante un ataque que pueda recibir la aplicación se ha realizado un árbol de ataque en el trato del componente, bloquear el robo de credenciales del sistema, punto que se especificado en la tabla 15. En la figura 17, se ha analizado a detalle si un usuario pierde el control en el acceso al sistema.

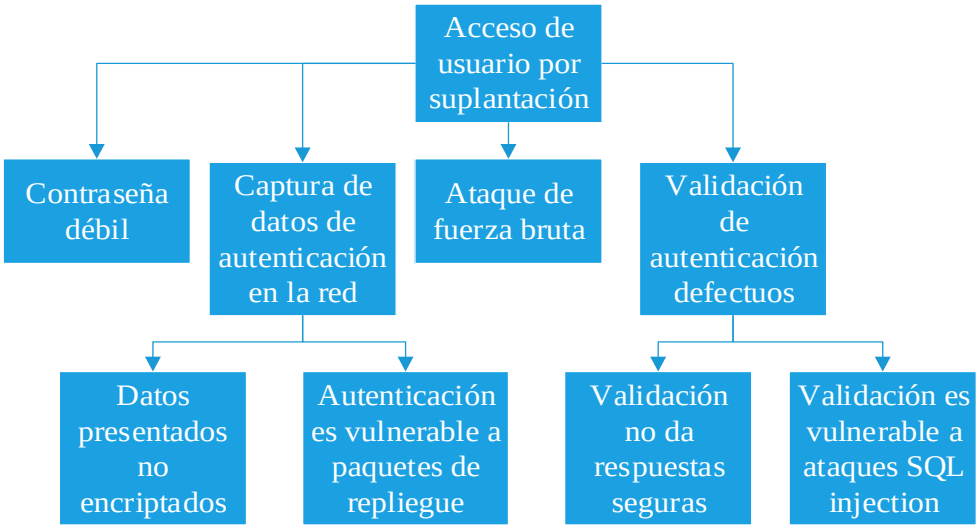

Figura 17. Ejemplo de árbol de ataque

De esta manera se ha trabajado en todos los requerimientos funcionales que debe cumplir la aplicación, es necesario tener en cuenta cómo se puede presentar el riesgo y sus posibles derivaciones. No obstante, no se puede llegar a determinar todas las amenazas que implica el riesgo, pero se puede tener la mayoría de amenazas o las más conocidas y con esto se puede generar una respuesta ante esos posibles riesgos.

3.2.6. Implementación

Para la fase de implementación segura se definió las herramientas con las cuales se implementó la aplicación como muestra la tabla 16, se lista las herramientas y su versión con las que se trabajó para el desarrollo del sistema.

Tipo de herramienta	Nombre	Versión
IDE	Netbeans	8.2
Framework de desarrollo	Yii2	2.0.14
Framework web (estilos)	Bootstrap	3.4
Manejador de dependencias	Composer	2.0.1
Control de versionamiento	Git	2.20.1
Visualización de base de datos	MySQL workbench	6.3.4
Cliente SFTP	WinSCP	5.13

Tabla 16. Herramientas aprobadas

Todas estas herramientas se han planificado conjuntamente con el equipo de desarrollo de la empresa BBM quienes realizan la aplicación principal y quienes han provisto de la función para el intercambio de información entre las dos aplicaciones.

Como parte de un procedimiento dentro de la fase de implementación se ha requerido hacer un análisis estático del código fuente en el cual se puede revisar técnicas de ejecución simbólica para el correcto uso de la codificación, como muestra la tabla 17, se puede tener diferentes capacidades de detección de problemas en las aplicaciones web, según Yeja (2016), establece que existen tres técnicas fundamentales para la detección de problemas las cuales son el análisis de flujo de datos, análisis semántico y análisis estructural

Técnica	Actividad	Revisión
Análisis semántico	División por cero	Si
	Uso de punteros nulos	Si
	Uso de mayor valor posible de una variable	Si
	Uso del menor valor de una variable	No
	Accesos fuera de rango en arreglos	Si
	Uso correcto de listas y arreglos vacíos	Si
	Comprobación de rangos de parámetros en las funciones	Si
Flujo de datos	No se leen variables sin no son declaradas	Si
	No se escriben variables más de una vez antes de ser leídas	Si
	No se escriben variables que después no son utilizadas	SI
Flujo estructural	Código accesible	Si
	Código anulado	No
	Bucles tienen condición alcanzable	Si

Tabla 17. Análisis estático

3.2.7. Verificación

Para esta fase se utiliza un análisis dinámico que puede ser utilizado para provocar errores en los programas introduciendo datos aleatorios sin tener en cuenta un formato establecido para este caso se ha realizado unos documentos en los cuales el usuario puede detallar los parámetros que se ingresaron y su respuesta obtenida.

El objetivo de realizar estas pruebas es establecer componentes específicos que resalten en este ambiente controlado. Como objetivo secundario es medir la infraestructura integrada ya que se utilizó un modelado de riesgos. Los usuarios se han categorizado en tres grupos en los cuales unos se han catalogados como blackbox user que examinan el sistema sin saber cómo funciona, los graybox user son los que tienes una referencia del funcionamiento del sistema y los whitebox user conociendo las políticas de seguridad instauradas y tiene un dominio y control de la información del sistema. Como muestra la tabla 18, se establece como es el formato con el cual los usuarios pueden reportar sus errores encontrados durante las pruebas de funcionalidad encontradas.

	PROCESO	GESTION DE DESARROLLO Y SOPORTE		
	FORMATO	REPORTE DE INCIDENTES EN DESARROLLO DE APLICACIONES WEB		
	CODIGO	Segundo proyecto-2019-001	**VERSIÓN**	1.22
FECHA Y HORA DEL REPORTE DE INCIDENTE				
LUGAR DONDE SE REPORTA EL INCIDENTE				
DESCRIPCION DE LOS INCIDENTES				
NUMERO DE INCIDENTE		**000**		
¿Qué Sucedió?: (Escriba una descripción de todas las acciones realizadas)				
Agregue una URL: (Ejemplo en el siguiente formato https://eldominio.com/action/)				
¿Qué error obtuvo?: (adjuntar una imagen del error presentado				

Observaciones adicionales:

Tabla 18. Reporte de incidentes con respecto a pruebas funcionales

Para la verificación de seguridad se realizan intrusiones en un ambiente controlado haciendo un símil a un hacking ético, donde se debe identificar los posibles inconvenientes que se presenten posteriormente a su entrega. Para esta prueba se ha utilizado la herramienta OWASP donde se ha utilizado en un modo ataque para simular un accionar similar en la vida real. La herramienta evalúa al menos los errores más comunes provistos por la Guía OWASP TOP 10 de riesgos en seguridad en aplicaciones web.

3.2.8. Liberación

Según Fulss (2015), como parte del proceso, esta fase debe ser ejecutada de dos a seis meses antes de la entrega del producto final al cliente. Sin embargo, por cuestión de los interesados se ha liberado una vez que se han realizado las pruebas funcionales y su respectiva corrección de errores. No obstante, se ha realizado un plan de respuesta ante incidentes de seguridad, el cual se detalla en la tabla 19, donde la especificación son actividades a realizar de acuerdo a un orden establecido.

Especificación	Orden
Supervisar los sistemas en busca de infracciones de seguridad.	0
Establecer comunicación para recibir los informes de incidentes de seguridad.	1
Documentar y catalogar los incidentes de seguridad.	2
Aumentar el nivel de conciencia con respecto a la seguridad dentro de la compañía para ayudar a evitar que se den incidentes en la organización.	3
Obtener más información sobre las nuevas vulnerabilidades y estrategias de ataque empleadas por los atacantes.	4
Investigar acerca de nuevas revisiones de software.	5
Analizar y desarrollar nuevas tecnologías para minimizar los riesgos y vulnerabilidades de seguridad.	6
Ofrecer servicios de consultoría sobre seguridad.	7
Perfeccionar y actualizar continuamente los sistemas y procedimientos actuales.	8

Tabla 19. Plan de respuesta ante incidentes

3.2.9. Respuesta

Con cada anuncio de incidentes se ha aplicado el respectivo plan de contingencia a incidentes detallado en la tabla 15. Con cada solución implementada se obtiene información relevante para ampliar la cobertura en cuanto a conocimiento y respuesta contra incidentes de seguridad.

3.3. Obtención de métricas

De acuerdo al criterio de Pressman & Troya (2019), la medición es un elemento clave en cualquier proceso de ingeniería, lo que permite entender mejor los atributos de los modelos que se crean y sirven para valorar la calidad de los productos.

Pressman & Troya (2010) afirman que las métricas del producto dentro de la ingeniería del software son imperfectas de acuerdo a un criterio físico, sin embargo, pueden proporcionar una forma sistemática de valorar la calidad del producto en conjunción de reglas definidas con claridad. Además, brinda una comprensión inmediata del entorno de desarrollo, lo cual permite descubrir y corregir potenciales problemas.

Definición de métricas

Las métricas de proceso y proyecto de software son medidas cuantitativas que permiten obtener comprensión acerca de la eficacia del proceso del software y de los proyectos que se realizan, usando el proceso como marco conceptual. Existen métricas durante todo el ciclo de vida, en este estudio se hace énfasis en las métricas para código, métricas para pruebas y métricas basadas en la función.

Métricas de código fuente

En el capítulo dos se ha definido las métricas de código fuente por lo que en este apartado se establece la métrica para el manejo del volumen de requerimientos con respecto al volumen de líneas de código generadas.

Métricas de pruebas

Se ha considerado las características de los dos proyectos y la experiencia de los profesionales que las elaboraron. Pressman & Troya (2010) define como principio de la métrica al punto de vista del usuario y el contexto de uso de la aplicación. En tal punto se ha categorizado en métricas de eficiencia y seguridad.

Métricas de Eficiencia

Es una métrica de calidad que proporciona beneficio tanto en el nivel del proyecto como en el del proceso y eficiencia de remoción del defecto. Para esta métrica se define el número de líneas de código respecto a bugs encontrados

Métricas de Seguridad

Las métricas de seguridad se aplican para la identificación de vulnerabilidades en un contexto a un objetivo determinado y en la cual se trabaja considerando el modelo del negocio y funcionamiento de la aplicación. Para esta métrica se define como el número de errores de seguridad respecto al número de líneas de código generadas, todo esto evalúa el nivel de riesgo que existe dentro de los proyectos.

Métricas de Productividad

Las métricas de productividad se aplican a la variable de estimación adecuada y se infiere el costo o esfuerzo. Las estimaciones de función se combinan para producir una estimación global para todo el proyecto. Para esta métrica se define como medida el tiempo para corregir bugs de programación y el tiempo para corregir errores de seguridad. Adicionalmente, se añaden como métrica el costo financiero que incurre el proyecto respecto a tiempo invertido como uso de recurso.

CAPÍTULO IV

COMPARACIÓN DE VIABILIDAD

Este capítulo presenta datos de comparación entre dos proyectos, el primero sin la aplicación de normativas relacionadas a codificación segura y el segundo aplicando la norma (ISO/IEC_27002, 2013) con soporte en la metodología SDL para la codificación segura, para determinar los beneficios en la elaboración de proyectos futuros dentro de una empresa pequeña que se dedica al desarrollo de aplicativos webs.

4.1. Método

Para el presente estudio se ha aplicado un método comparativo de investigación en el cual se puede evaluar si al utilizar la norma 27002:2013, como antecedente de un proyecto, ayuda a la creación de aplicaciones web, el método también permite comparar la situación más favorable que existe entre un proyecto en relación con otros proyectos similares, enfatizando en el riesgo obtenido, adicional se puede comparar en que situaciones existe menor riesgo de seguridad y decidir cuál ha sido la opción más fructífera en el transcurso del proyecto.

4.1.1. Para las encuestas de validación

La búsqueda de información y recolección de datos se realiza en base a los elementos del problema para obtener la mejor decisión para el desarrollo de proyectos en aplicaciones web utilizando la normativa 27002, con ayuda de una herramienta que brinde una guía en la implementación de codificación segura.

4.1.1.1. Población y muestra

Para los dos proyectos se trabajan con todo el equipo de desarrollo lo que significa la totalidad del universo, como detalla la tabla 20.

Población	Número	Porcentaje
Equipo de desarrollo primer proyecto	6	100%
Equipo de desarrollo segundo proyecto	6	100%

Tabla 20. Población de estudio

4.1.1.2. Recolección de información

La recolección de datos se realizó en las instalaciones de la empresa, el proceso de recolección de información se ejecutó basándose en encuestas de validación sobre temas puntuales, donde se evaluó los dominios A5 políticas de seguridad de la información, A6 Organización de la seguridad de la información, A8 Gestión de control de acceso, A10 criptografía, A11 Seguridad física y del entorno, A12 Seguridad de las operaciones, A13 Seguridad de las comunicaciones, y como punto central en A14 Adquisición, desarrollo y mantenimiento de los sistemas de información.

Para la elaboración de la técnica se utilizaron 123 enunciados de cumplimiento de los controles respectivamente a cada dominio, su nivel de cumplimiento se ha expresado en dos formatos: si cumple (SI) o no cumple (NO).

La primera encuesta se la realizó previo el inicio del segundo proyecto, debido a que para nuestro análisis es necesario establecer un parámetro de cumplimiento de normas de seguridad, y que en ese momento no cuenta con la implementación de una normativa de seguridad dentro de la empresa. La segunda, se realizó al finalizar el segundo proyecto que fue elaborado aplicando la norma ISO 27002:2013 y con soporte de herramientas que brindan una codificación segura, como nota aclaratoria, se realizaron las mismas preguntas en ambas encuestas.

4.1.1.3. Procesamiento y análisis de datos

Se ha realizado una revisión crítica de la información recolectada; es decir limpieza de información defectuosa, incompleta, no pertinente y otras fallas. Las encuestas se han

tabulado en cuadros según el cumplimiento de la norma. El manejo de la información se ha elaborado un reajuste de cuadros con casillas vacías por lo que no influyen significativamente en los análisis. Para finalizar se ha elaborado un estudio estadístico de datos para presentación de resultados con respecto a cada dominio de control.

4.1.2. Para las métricas

Según Serrano, Piattini, Calero, Miranda y Alarcos (2002) las métricas debe basarse en objetivos de medida claros y siguiendo las necesidades de la organización. Sin embargo, según Zubrow (2004), las métricas de software se acomodan a la necesidad de la validación, no hay precedente que integre los aspectos que son necesarios para dicha validación, de tal manera en este estudio se ha utilizado las siguientes métricas que se detallan a continuación.

4.1.2.1. Definición de métricas

Los parámetros para medir se han establecido de acuerdo a métricas de pruebas y de código, para mayor visibilidad se muestra en la tabla 21 donde: *tipo* es la métrica donde debe ser aplicada, *relación* describe variables que se pueden comparar y el *resultado* es el valor que se obtiene al aplicar la relación entre variables.

Tipo	Relación	Resultado
Código	Requerimientos / línea de código generada	Porcentaje
Pruebas	Eficiencia Bugs encontrados / líneas de código	Porcentaje
Pruebas	Seguridad Fallos de seguridad / líneas de código	Porcentaje
Productividad	Errores cometidos (seguridad - codificación) / tiempo empleado	Número de horas
Productividad	Tiempo invertido / costo de proyecto	Horas

Tabla 21. Métricas a obtener

4.1.2.2. Población y muestras

El universo de la población para cada proyecto es distinto por eso se ha elaborado un detalle de su caracterización como muestra la tabla 22.

Población	Universo	Muestra	Porcentaje
Grupo de usuario de pruebas funcionales primer proyecto	60	6	10%
Grupo de usuario de pruebas funcionales segundo proyecto	3000	25	0,84%

Tabla 22. Población para ejecución de métricas

4.1.2.3. Definición de interacción

Para la creación de la prueba se han tomado en cuenta los siguientes aspectos, como principal elemento se ha tomado que la funcionalidad del sistema sea correcta, en los cuales se ha creado una matriz de requerimientos de una prueba donde existe un proceso de funcionalidades a probar como muestra la tabla 23.

Funcionalidad	Cumplimiento	
	SI	NO
Identificador del requerimiento de prueba	x	
Descripción del requerimiento de prueba		x
Datos de entrada o elemento a probar	x	
Resultado esperado q Iteración de la prueba	x	
Observaciones	x	
Versión	x	

Tabla 23. Proceso de funcionalidades a probar

4.1.2.4. Preparación ambiente de pruebas

Para pruebas de funcionalidad se ha definido la configuración de máquinas (clientes y servidores) en que se ejecutarán las pruebas; el sistema operativo, browser y configuración TCP/IP.

Para pruebas de seguridad se ha elegido una herramienta que evalúe los criterios de seguridad más populares como es OWASP Top 10 (2017), esta guía ayuda a la caracterización del entorno dentro de la configuración de la herramienta, por ejemplo una de sus opciones es el modo ataque la cual evalúa a mayor profundidad los criterios de seguridad de la guía y provee un detalle de los errores de seguridad que la aplicación posee durante su análisis con esta herramienta.

4.1.2.5. Recolección de información

La recolección de información se ha realizado una compilación de todas las pruebas de funcionalidad de los usuarios con los cual se han eliminado casos repetidos, o que no pertenecen a la definición de las métricas.

Con respecto al número de líneas de código se ha elaborado un script en el que recoja el número de líneas elaboradas los archivos pertinentes al modelo MVC

El tiempo se ha medido mediante una plataforma de cronometraje que se usa para todos los proyectos dentro de la empresa

4.1.2.6. Procesamiento y análisis de los datos

Se ha elaborado un estudio estadístico de datos para presentación de resultados con respecto a cada métrica presentada en relación a los dos proyectos.

4.2. Resultados

4.2.1. Aplicación de ISO 27001

En base a la tabla 20 se puede mencionar que se ha tomado en cuenta los 14 dominios y se han evaluado los 123 controles tomados del Anexo A de la norma ISO 27001, cabe recalcar que para el primer proyecto no se tenía en consideración la normativa para la implementación del proyecto, sin embargo, se ha evaluado con todos los controles como se aprecia en la tabla 24 se ha establecido el cumplimiento de cada control.

Dominio	Nivel de cumplimiento de controles	Primer proyecto	Segundo proyecto
Políticas de seguridad de la información	SI	87,50%	87,50%
	NO	12,50%	12,50%
Organización de la seguridad de la información	SI	43,18%	59,09%
	NO	56,82%	38,64%
Seguridad relativa a los recursos humanos	SI	66,67%	66,67%
	NO	44,44%	44,44%
Gestión de activos	SI	61,67%	65,00%

Dominio	Nivel de cumplimiento de controles	Primer proyecto	Segundo proyecto
Control de acceso	NO	38,33%	38,33%
	SI	26,14%	61,36%
Criptografía	NO	72,73%	38,64%
	SI	37,50%	50,00%
Seguridad física y del entorno	NO	62,50%	50,00%
	SI	36,90%	57,14%
Seguridad de las operaciones	NO	63,10%	42,86%
	SI	37,50%	66,67%
Seguridad de las comunicaciones	NO	61,46%	32,29%
	SI	56,82%	79,55%
Adquisición, desarrollo y mantenimiento de los sistemas de información	NO	43,18%	20,45%
	SI	43,18%	75,00%
Relación con proveedores	NO	56,82%	23,53%
	SI	57,14%	85,71%
Gestión de incidentes de seguridad de la información	NO	42,86%	14,29%
	SI	47,50%	60,00%
Aspectos de seguridad de la información para la gestión de la continuidad de negocio	NO	52,50%	30,00%
	SI	57,14%	100,00%
Cumplimiento	NO	42,86%	0,00%
	SI	38,46%	69,23%
	NO	61,54%	30,77%

Tabla 24. Cumplimiento con respecto a la norma 27001

Dentro de los aspectos de evaluación de cada control se puede encontrar tanto documentación que hace referencia a un control o la implementación del control dentro del Sistema de Gestión de la información (EGSI) de la empresa, cabe aclarar que según Peñaherrera (2013), el EGSI no establece la implementación de un sistema de seguridad de la información, sino que busca una correcta gestión de la seguridad de la información mediante directrices contenidas en la norma.

La figura 18 muestra de manera generalizada los resultados en comparación del primer proyecto en relación con el segundo proyecto muestran que se tiene una mejora sustancial, sin embargo, no llega a tener un manejo optimo en los aspectos de seguridad, en

algunos dominios de control esta mejora es inicial y es necesario trabajar más mantener una mejora continua para futuros proyectos.

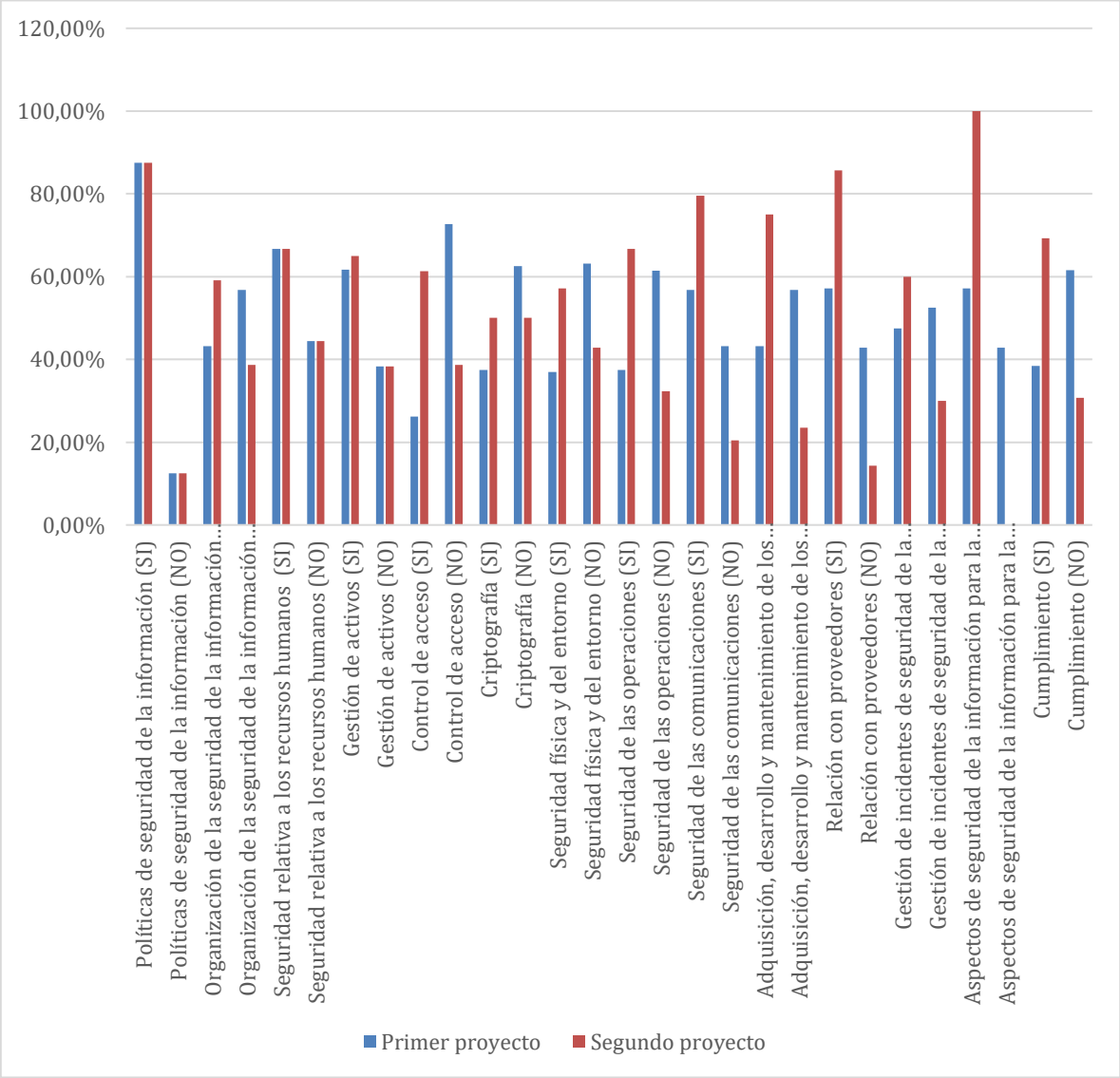

Figura 18. Cumplimiento de norma ISO 27001

4.2.2. Métrica de requerimientos vs LOCs

Como parte del cumplimiento de los requerimientos y tomando en cuenta su validación y tomando en cuenta la complejidad y el esfuerzo se ha elaborado la tabla 25 donde se describe los requerimientos y las líneas de código.

Enunciado	Primer proyecto	Segundo proyecto
Requerimientos	10	19
Líneas de código	23970	24422
Errores	234	170
%	0,04171882	0,07779871

Tabla 25. Detalle de requerimientos en relación a líneas de código y errores encontrados.

De una forma global en la tabla 21 se puede apreciar que el segundo proyecto tiene mayores requerimientos a cumplir a diferencia del primer proyecto, sin embargo, el número de líneas de código generadas no difiere mucho en los dos proyectos, esto se debe a que el segundo proyecto se ha implementado con la norma ISO 27001, la metodología SDL y en la parte de codificación segura la guía del desarrollador de OWASP, esto refleja que se ha optimizado el uso de recursos durante la implementación del segundo proyecto.

4.2.3. Métrica errores de programación vs líneas de código

Dentro de las principales acciones que se toman para la codificación segura se hace como prioritario el manejo y control de los accesos a usuarios, la protección de los datos, y el uso de memoria.

Para encontrar los errores de programación y medir la eficiencia de los dos proyectos se han realizado pruebas de caja negra y funcionales a su universo como se detalla en la tabla 18. Cabe mencionar que la forma recopilar los datos para el segundo proyecto cambió ya que se utilizó como fuente de reporte de errores a WhatsApp y Skype, no obstante, la compilación de los datos ha sido el mismo para los dos proyectos.

Después de la compilación de los datos el número de errores de programación ha disminuido con respecto a las líneas de código generadas como muestra la figura 19.

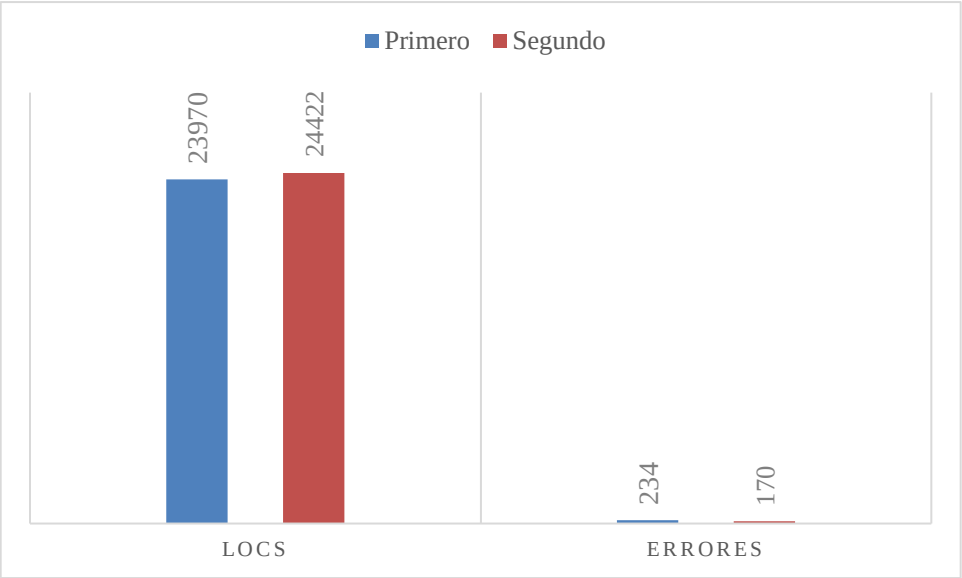

Figura 19. Líneas de código respecto a errores de programación

Como parte del estudio a los errores de programación se los ha categorizado en tres niveles de criticidad los cuales son bajo, medio, alto, que miden el impacto que mantuvieron durante la ejecución de las pruebas funcionales. En la figura 20 se observa con claridad que los errores de índice alto, así como los errores de índice medio han sido reducidos drásticamente en comparación de los dos proyectos.

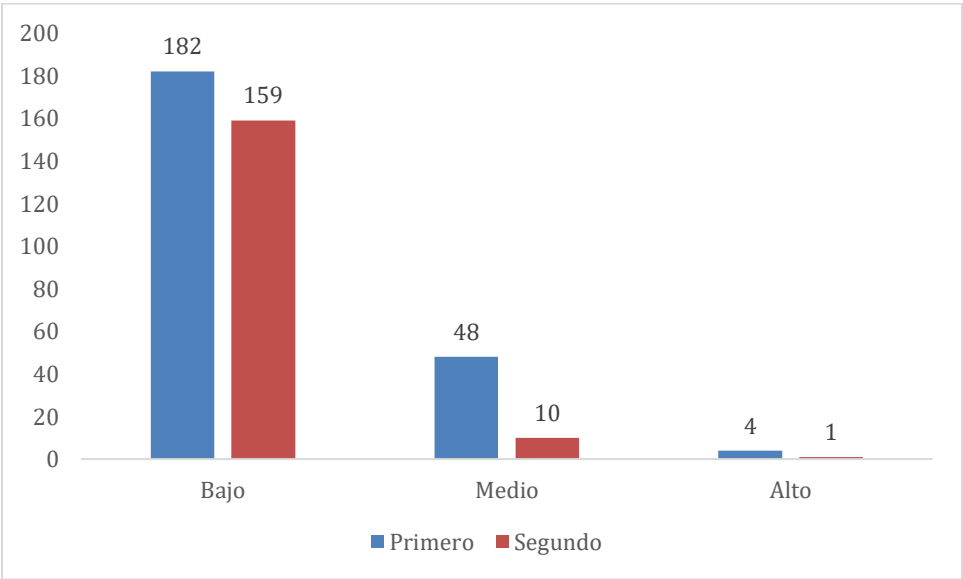

Figura 20. Nivel de criticidad de los errores

En la tabla 26 se puede observar que, aunque las líneas de código sean mayores su error representado es menor.

Enunciado	Primer proyecto	Segundo proyecto
Líneas de código	23970	24422
Errores	234	170
%	0,97622028	0,69609369

Tabla 26. Errores de código vs líneas de código generadas

4.2.4. Métrica número de errores de programación vs tiempo invertido en solución

Como muestra la tabla 27 el tiempo empleado en la corrección de errores depende del número de incidencias y la gravedad del error, para lo cual se establece que entre mayor sea el nivel de criticidad del error será mayor el número de horas a emplear.

Horas invertidas	Bajo	Medio	Alto	Proyecto
95	56	27	12	Primer
35	28	6	1	Segundo

Tabla 27. Tiempo empleado para corregir en relación al nivel de riesgo

De forma global en la tabla 27, se puede observar que para el primer proyecto se utiliza más horas/hombre para la corrección de errores de programación a diferencia del segundo proyecto, esto prima porque se han aplicado técnicas de codificación segura para el segundo proyecto, lo que muestra que se obtiene menor número de errores de programación.

4.2.5. Métrica fallos de seguridad vs líneas de código

En el estudio se ha utilizado OWASP ZAP como herramienta de evaluación de seguridad, la cual ayuda a encontrar automáticamente vulnerabilidades de seguridad en sus aplicaciones web mientras se ejecuta la prueba en las aplicaciones.

Para la medición de fallos de seguridad hay que mencionar las etapas en las cuales se elaboraron las pruebas, para el primer proyecto se ha realizado la prueba en la fase de mantenimiento, y para el segundo proyecto se ha realizado al término de la fase de desarrollo, OWASP ZAP se configuró en modo de ataque para que pueda encontrar el mayor número de vulnerabilidades que se pueda suscitar dentro de los dos sistemas

En la Figura 21, se puede observar un ejemplo de fallos de seguridad obtenidos durante la ejecución de la prueba de seguridad para el primer proyecto

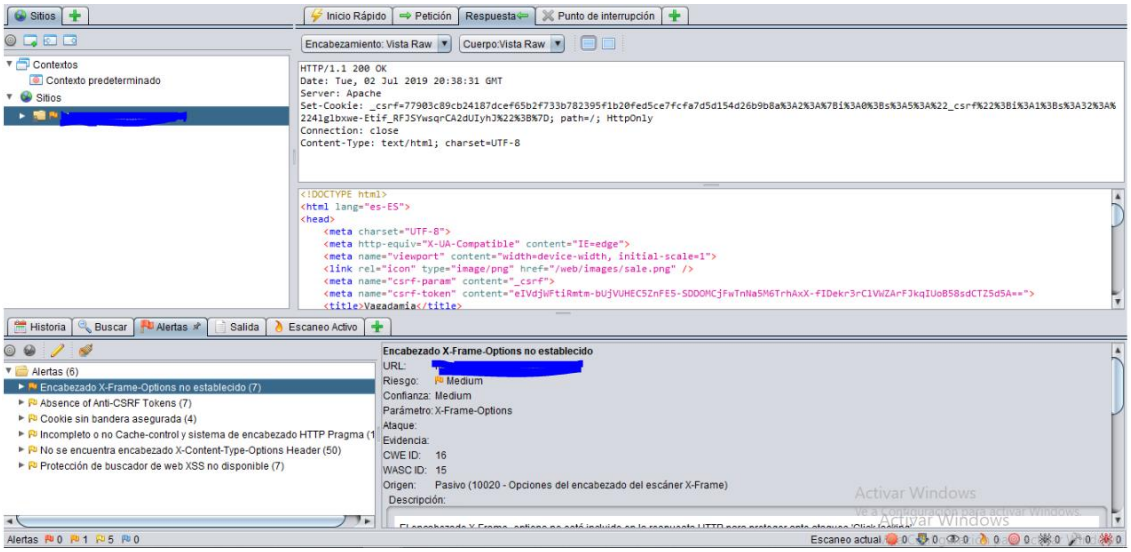

Figura 21. Ejemplo de muestra de errores obtenidos utilizando OWASP ZAP

En la Figura 22, se pueden observar un ejemplo de fallos de seguridad obtenidos durante la ejecución de la prueba de seguridad para el segundo proyecto.

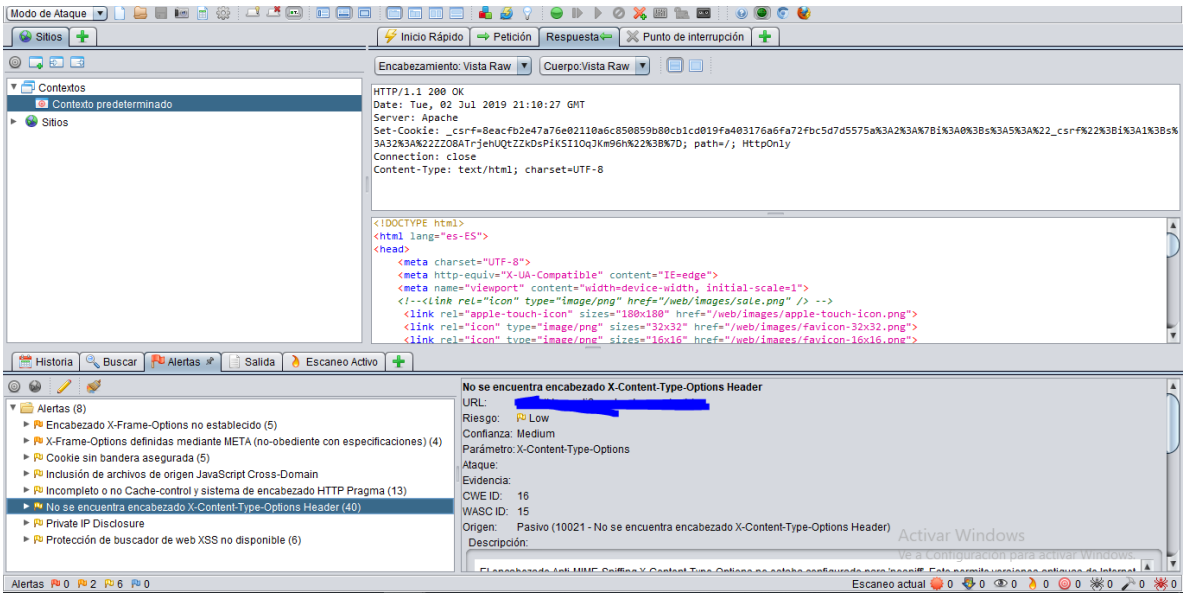

Figura 22. Ejemplo de muestra de errores obtenidos utilizando OWASP ZAP

Una vez terminado las pruebas de seguridad se ha compilado los errores y se ha categorizado por su nivel de riesgo tal y como muestra la tabla 28.

Riesgo	Primer proyecto	Segundo proyecto	Nivel de riesgo
Encabezado X-frame-options	7	5	Medio
Ausencia anti-CSRF tokens	7	0	Medio
Cookie sin bandera	4	5	Bajo
No cache control	16	13	Bajo
Encabezado X-Content-Type-Options	50	40	Bajo
Protección de web XSS	7	6	Bajo
Javascript Cross-domain	1	0	Medio
Revelación de IP privada	1	0	Bajo
Total	93	69	

Tabla 28. Errores encontrados por nivel de criticidad

Aunque existe alertas de seguridad no se ha detectado ningún riesgo de nivel alto para ninguno de los dos proyectos, esto significa que el primer proyecto, aunque no se conoce los estándares de codificación segura mantiene cierto nivel de seguridad con respecto al segundo proyecto, sin embargo, en los dos casos se encuentran errores con riesgo medio que implica un indicio para posibles ataques e intromisiones en el aplicativo web. El segundo proyecto se realizó utilizando codificación segura, no obstante, se han encontrado algunos errores de riesgo medio, que sin duda se pueden corregir de manera oportuna en la fase de mantenimiento.

4.2.6. Métrica número de fallos de seguridad vs tiempo invertido en solución

De acuerdo al nivel de riesgo el número de horas invertidas para corregir estos fallos se seguridad es mayor, con esta premisa en la tabla 29 detallamos el número de horas invertidas para corregir los fallos en los dos proyectos.

Riesgo	Primer P/ Hora	Segundo P / Hora	Nivel de riesgo
Encabezado X-frame-options	2	2	Medio
Ausencia anti-CSRF tokens	2		Medio
Cookie sin bandera	1	1	Bajo
No cache control	1	1	Bajo
Encabezado X-Content-Type-Options	2	1	Bajo
Protección de web XSS	1	1	Bajo

Riesgo	Primer P/ Hora	Segundo P / Hora	Nivel de riesgo
Javascript cross domain	3	0	Medio
Revelación de IP privada	1	0	Bajo
Total	13	6	

Tabla 29. Tiempo empleado para corrección de errores de seguridad por nivel de criticidad

Cabe mencionar que el uso de normas y metodologías de desarrollo seguro ayudan a identificar de mejor manera los fallos y su categorización guía de manera eficiente a la corrección de los mismo lo que implica que el número de horas se reduce.

4.2.7. Métrica tiempo invertido vs costo de proyecto (Productividad)

La asignación de recursos como horas invertidas en el desarrollo, personal calificado asignado, insumos y servicios durante la implementación del proyecto hace que el costo de un proyecto sea mayor respecto a otro, por esta razón se ha analizado los costos del proyecto.

En la tabla 30 se ha elaborado un detalle de horas invertidas durante cada fase del ciclo de vida de cada proyecto.

Fase	Primer proyecto	Segundo proyecto
Estudio de Normativa ISO 27001 - ISO 27002		35
Aplicación de métricas Anexo A ISO 27001		40
Requerimientos	33	125
Análisis y requerimientos y diseño	52	89
Implementación y codificación	222	313
Pruebas	12	26
Mantenimiento	95	35
Total	414	663

Tabla 30. Detalle de horas invertidas en cada proyecto

En la tabla 31 de detalla el número de horas invertidas para cada proyecto y se ha incluido el valor de la hora, este parámetro ayuda para evidenciar que no para todos los proyectos el costo de hora es el mismo ya que depende de cliente al que se le va implementar el sistema. Esto depende mucho de las políticas de la empresa.

Proyecto	Horas	Valor $	Costo del proyecto	Equipo
Primer proyecto	380	20	7600	4
Segundo proyecto	640	30	19200	4

Tabla 31. Pronóstico de horas y costo de proyecto

4.3. Discusión

Se han encontrado algunos criterios optimistas con respecto al estudio. Aunque la aplicación de una normativa de seguridad en los dos proyectos es diferente, el caso del primer proyecto ayudó a comprender la forma de aplicar las contramedidas que se pueden implementar en la codificación segura utilizando la norma ISO 27001:2013 y la guía de desarrollador de OWASP. Además, se puede representar una función que señala las relaciones del conocimiento de seguridad y herramientas en el ciclo de vida, lo cual es útil para el desarrollo de codificación segura con respecto al tiempo invertido para el desarrollo de los dos proyectos, por lo que esta información es útil para la recopilación de conocimiento para la toma de decisiones para futuros proyectos.

Con respecto al valor total de cada proyecto, la implementación de la norma requiere costos adicionales para asociar las herramientas necesarias para realizar un proyecto mediante el desarrollo de codificación segura. Además, en la implementación del segundo proyecto, la norma es administrada por primera vez, por lo tanto, como la asociación de conocimiento de errores de seguridad se lleva a cabo por proyecto, la comparación de los proyectos muestra una diferencia en los errores de seguridad obtenido, esto es de acuerdo a la manera en la cual se haya codificado u ocasionalmente se haya aplicado medidas de seguridad al entorno donde se realizó el proyecto. Los elementos que componen el ciclo de vida del desarrollo del software están asociados con cada pieza de conocimiento de ingeniería de software.

A partir de la implementación de un proyecto con codificación segura respecto a otro que no se lo ha hecho, se puede confirmar que la efectividad de la aplicación de codificación segura a un entorno de desarrollo de un proyecto es efectiva, ya que el número de errores de seguridad encontrados son menores y el tiempo empleado para aplicar correctivos de seguridad ha reducido. Asimismo, el tiempo durante todo el ciclo de vida del proyecto es

menor en relación al primer proyecto sin tomar en cuenta el estudio de normas de seguridad y su respectiva aplicación. Además, se encontró que la utilización de una metodología ágil, la segmentación de roles, asignación de funciones para el personal, y la aplicación de una guía de operación dinámica y efectiva con respecto al control de cambios conjuntamente con herramientas de conocimiento de seguridad es útil para el aprendizaje y toma de decisiones para futuros proyectos.

En relación a las horas reales con respecto al costo en el primer proyecto existe un 8,94% de excedente de tiempo que sobrepasa al estimado al inicio del mismo. El tiempo invertido en cada una de las fases del ciclo de vida del segundo proyecto se detalla de la siguiente manera: para requerimientos 7,97%, para análisis y diseño 12,56%, para implementación y codificación 53,62%; como nota curiosa se puede demostrar que para codificar se lleva más de la mitad del tiempo de todo el proyecto, para pruebas 2,89% y para mantenimientos el 22,94% de las horas para el proyecto. Como se puede evidenciar existe un número alto de horas en la fase de mantenimientos que representa un 22,94%, el cual ha sido empleado para agregar algunas funcionalidades que no se mencionaron en el documento de requerimientos. Como muestran los resultados al implementar el aseguramiento del proceso del ciclo de vida mejora el tiempo de ejecución del producto.

CAPÍTULO V

CONCLUSIONES Y TRABAJOS FUTUROS

5.1. Conclusiones

De forma global tanto en el primer proyecto como en el segundo proyecto se han utilizados técnicas y metodologías asociadas al correcto desarrollo de aplicativos webs, como por ejemplo se ha escogido un modelo de ciclo de vida de desarrollo ágil (Xtreme programming) para los dos proyectos, el mismo que enfatiza la satisfacción del cliente, ya que en ningún caso se han producido retrasos en la entrega y se ha mantenido reuniones frecuentes con el cliente para obtener la solución a sus objetivos y la mejor medida de cada uno de los proyectos ha sido ver funcionando el aplicativo de acuerdo a los requerimientos y lo que el cliente desea obtener. Adicionalmente, para el segundo proyecto se ha implementado la metodología DSL que brinda aseguramiento a todo el ciclo de vida, lo cual ayuda al trabajo de equipo donde se incluye administradores, desarrolladores e interesados convirtiéndose en un entorno simple, seguro y altamente productivo.

Para el segundo proyecto se ha utilizado una norma (ISO/IEC_27001, 2013) la cual ofrece un control en cada área de seguridad dentro de la empresa y para el desarrollo de sistemas de información. Una muestra evidente entre los dos proyectos es el control de cambios, utilizando la metodología ágil simplemente se plasmaría en un comentario dentro de la programación, no obstante, cuando se aplica la norma ISO 27001 es necesario establecer un procedimiento que se adapte de mejor manera a la metodología ágil para el control de los cambios en los sistemas.

A relación entre los dos proyectos, el primer proyecto no ha tomado en cuenta los principios de codificación segura, a comparación del segundo proyecto que toman en consideración principios de seguridad tales como confidencialidad, integridad y disponibilidad, el hecho que se haya tomado en cuenta los principios de codificación segura

hace que el segundo proyecto sea más robusto en aspectos de seguridad y calidad manejando apropiadamente los errores ya que cuenta con una planificación de control y respuesta ante posibles errores de seguridad.

En el presente estudio se puede afirmar que sin la utilización de herramientas que permita visualizar, errores de seguridad, no se puede asegurar que un aplicativo sea 100% seguro, como menciona Verdon (2017), en su guía de desarrollo no se debe confiar en ningún valor sin estar seguro que ese valor no haya sido un agente de manipulación.

Después del estudio realizado y la aplicación de la norma ISO 270012013, siguiendo y aplicando la metodología SDL y ejecutando la guía de desarrollador provista por OWASP se puede considerar que el tiempo utilizado en la realización del segundo proyecto ha sido mucho mayor con respecto al primer proyecto, no obstante, se puede concluir que esto es resultado del estudio e implementación en conjunto de normas y metodologías.

5.2. Trabajos futuros

Con el presente estudio y el detalle de sus características ofrece nuevas ideas para contribuir en iniciativas que respalden o invaliden las afirmaciones que se presenta. De esta manera pueden surgir nuevos trabajos en consecuencia a los resultados de este estudio:

- Utilizando proyectos con similares peculiaridades y requerimientos donde su nivel de madurez en la aplicación de la normativa en un proyecto sea inicial y otro que mantenga el nivel de madurez en la aplicación de la norma en un estado repetible, con esto se puede evaluar en cuál de los dos niveles de madurez se puede encontrar mayor número de riesgos de seguridad o si la diferencia es muy amplia entre los dos proyectos.

- Utilizando una metodología de desarrollo seguro en el ciclo de vida diferente a DSL, se puede obtener resultados similares en el aspecto de seguridad con respecto a un proyecto que no haya utilizado codificación segura.

- Utilizando normas y estándares de seguridad de desarrollo de software o ejemplo ISO 31000 manejo de riesgos SDT o utilización de NIST cyber Framework y verificar si se obtienen los mismos riesgos de seguridad que en el presente estudio.

Bibliografía

Ambler, S. (2004). *The object primer: agile modeling-driven development with UML 2.0*. Cambridge University Press.

Berzal, F. (2004). *El ciclo de vida de un sistema de información*. Retrieved from http://flanagan.ugr.es/docencia/2005-2006/2/apuntes/ciclovida.pdf

Boehm, B. (1988). A Spiral Model of Software Development and Enhancement. *Computer*, (5), 61–72.

Calabrese, J., Muñoz, R., Pasini, A., Esponda, S., Boracchia, M., & Pesado, P. (2017). Asistente para la evaluación de características de calidad de producto de software propuestas por ISO/IEC 25010 basado en métricas definidas usando el enfoque GQM. *XXIII Congreso Argentino de Ciencias de La Computación (La Plata, 2017)*, pp. 660–671.

Callejas, M., Alarcón, A., & Álvarez, A. M. (2014). Metodologías y procesos de análisis de software. *Repositorio Académico*, *13*(1), 35–50.

Castellaro, M., Romaniz, S., Ramos, J. C., Feck, C., & Gaspoz, I. (2016). Aplicar el Modelo de Amenazas para incluir la Seguridad en el Modelado de Sistemas. *In V Congreso Iberoamericano de Seguridad Informática-CIBSI*, 16.

Castellaro, M., Romaniz, S., Ramos, J., & Pessolani, P. (2009). Hacia la Ingeniería de Software Seguro. *XV Congreso Argentino de Ciencias de La Computación*, *610*, 10.

Cataldi, Z. (2000). *Una metodología para el diseño, desarrollo y evaluación de software educativo (Doctoral dissertation, Facultad de Informática).* Universidad Nacional de La Plata.

Cova, Á., Arrieta, X., & Aular De Duran, J. (2008). Revisión de modelos para evaluación

de software educativos (Revision Of Models To Evaluate Educative Software). *Télématique*, *7*(1), 93–116.

Cutanda, D. (2010). *Integración de la herramienta SGSI Tracking con la norma ISO 27001 y el reglamento de la LOPD en un entorno virtualizado*. Universidad Politécnica de Valencia.

Davis, N., Humphrey, W., Redwine, S. T., Zibulski, G., & McGraw, G. (2004). Processes for producing secure software. *IEEE Security & Privacy Magazine*, *2*(3), 18–25.

De Win, B., Scandariato, R., Buyens, K., Grégoire, J., & Joosen, W. (2009). On the secure software development process: CLASP, SDL and Touchpoints compared. *Information and Software Technology*, *51*(7), 1152–1171.

DeMarco, T. (1979). Structure analysis and system specification. In *Pioneers and Their Contributions to Software Engineering* (pp. 255–288). Springer.

Diaz, L. (2015). *Desarrollo de software seguro (Msc thesis)*. Universidad Piloto de Colombia.

Enríquez, L., Farías, E., Flores, E., Honores, C., Llanos, R., Soto, G., & Zúñiga, A. (2017). *Metodologia de desarrollo de software*. Cimbote: Universidad Catolica Los Angeles Chimbote.

Fulss, D. C. (2015). Técnicas y herramientas para el control de procesos y la gestión de la calidad, pasa su uso en la auditoría interna y en la gestión de riesgos. *Consejo de Auditoría Interna General de Gobierno*, 163.

Gane, C., & Sarson, T. (1977). *Structured Systems Analysis: Tools and Techniques, Improved Systems Technologies*. New York.

García, A., & Alegre, M. del P. (2011). *SEGURIDAD INFORMATICA ED. 11 Paraninfo*. 11.

Ghani, I., & Yasin, I. (2013). Software Security Engineering in Extreme Programming

Methodology: A Systematic Literature Review. *Science International*, *25*(2), 215–221.

Gonzales, H. (2001). onceptos básicos de Métricas. *Las Metricas de Software y Su Uso En La Region*, 6–14.

Gonzalez, H. (2001). Métricas en el desarrollo del Software. *Las Metricas de Software y Su Uso En La Region*, 60–101.

Halstead, M. (1977). Elements of Software Science. 1977 (Operating and programming systems series). *Out of Print*.

Henderson-Sellers, B., & Edwards, J. M. (1990). The object-oriented systems life cycle. *Communications of the ACM*, *33*(9), 142–159.

Hernández, A. (2012). *Medidas de productividad en los proyectos de desarrollo de software: una aproximación por puestos de trabajo (Doctoral dissertation*. Universidad Carlos III de Madrid.

Highsmith, J. (2013). *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems* (Addison-Wesley, Ed.).

ISO/IEC_27001. (2013). INFORMATION SECURITY MANAGEMENT SYSTEMS. *Security Techniques*, 23.

ISO/IEC_27002. (2013). CODE OF PRACTICE FOR INFORMATION SECURITY CONTROLS. *Security Techniques*, *2*, 80.

Keramati, H. (2008). Integrating Software Development Security Activities with Agile Methodologies. In *IEEE/ACS International Conference on Computer Systems and Applications* (pp. 749–754). IEEE.

Kroll, J., Fontoura, L. M., Wagner, R., & Ornellas, M. C. D. (2008). Usando Padrões para o Desenvolvimento da Gestão da Segurança de Sistemas de Informação baseado na Norma ISO / IEC 21827 : 2008. In *Simpósio Brasileiro de Sistemas de Informação*. SBSI.

Lehman, M., Ramil, J., Wernick, P., Perry, D., & Turski, W. (1997). Metrics and laws of software evolution-the nineties view. In *Proceedings Fourth International Software Metrics Symposium*. Albuquerque: IEEE.

Letelier, P., Canós, M., Sánchez, E., & Penadés, M. (2003). Métodologías Ágiles en el Desarrollo de Software. *Taller Metologías Ágiles En El Desarrollo de Software.*, 1–8. Retrieved from http://www.carlosfau.com.ar/nqi/nqifiles/XP_Agil.pdf

Letelier, P., Penadés, C., Canós, J., & Sánchez, E. (2012). *Metodologías Ágiles en el Desarrollo de Software*. 59.

Maña, A., Ray, D., Sánchez, F., & Yagüe, M. I. (2004). Integrando la Ingeniería de Seguridad en un Proceso de Ingeniería Software. *VIII Reunión Espñola de Criptología y Seguridad de La Información, RECSI*, (May 2014), 383–392.

Martinez-garcia, H. A. (2013). *Análisis de Riesgos y Amenazas en el Modelo de Cómputo en la Nube para Alojamiento de Respaldos de Datos*.

McGraw, G. (2006). *Seven Touchpoints for Software Security* (Vol. 1). Addison-Wesley Professional.

Ministerio de Educación. (2018). *Currículo de los niveles de educación obligatoria*. *2*, 13–230.

Peñaherrera, C. C. (2013). *Esquema gubernamental de seguridad de la informacion egsi*. 1–47.

Pesado, P., Bertone, R. A., Esponda, S., Pasini, A. C., Boracchia, M., Martorelli, S., & Swaels, M. (2013). Mejora de procesos en el desarrollo de sistemas de software y en procesos de gestión. *XV Workshop de Investigadores En Ciencias de La Computación*, *15*(1), 581–585.

Pressman, R., & Troya, J. (2010). Ingeniería Del Software I. In *Ingeniería Del Software I*. McGraw Hill.

Rindell, K., Hyrynsalmi, S., & Leppanen, V. (2015). A comparison of securityassurance support of agile software development methods. In *Proceedings of the 16th International Conference on Computer Systems and Technologies* (Vol. 3, pp. 31–68). ACM.

Rose, K. (2013). A Guide to the Project Management Body of Knowledge (P M B O K ® G U I D E). *Project Management Journal*, *44*(3), 157–201.

Rumbaugh, J., Blaha, M., Premerlani, W., & Lorensen, W. (1991). *Object-oriented modeling and design*. Englewood Cliffs New Jersey: Prentice-hall.

Sánchez, Á. (2003). *Diseño de un Sistema de Gestión de la Seguridad de la Información para Comercio Electrónico basado en la ISO 27001 para pequeñas y medianas empresas en la ciudad de Quito(Bachelor's thesis)*. Pontificia Universidad Católica del Ecuador.

Schwaber, K., & Beedle, M. (2002). *Agile software development with Scrum*. Prentice Hall Upper Saddle River.

SERCOP. (2017). *Asignación de órdenes de compra a proveedores catalogados para la adquisición de bienes y servicios normalizados del Catálogo Electrónico*.

Serrano, M., Piattini, M., Calero, C., Genero, M., Miranda, D., & Alarcos, G. (2002). Un método para la definición de métricas de software. *1er Workshop En Métodos de Investigación y Fundamentos Filosóficos En Ingeniería Del Software y Sistemas de Información*, 65–74.

Sommerville, I. (2005). *Ingeniería del software.* (p. 712). p. 712. Pearson educación.

Stohlman Jr., F., & Brecher, G. (2013). Humoral regulation of erythropoiesis III. Effect of exposure to simulated altitude. *The Journal of Laboratory and Clinical Medicine*, *49*(6), 890–895.

Sun, G., Yajima, K., Miura, J., Shi, K., Goto, Y., & Cheng, J. (2012). A supporting tool for creating and maintaining security targets according to ISO/IEC 15408. *In 2012 IEEE*

International Conference on Computer Science and Automation Engineering, 745–749.

Tejas, A. H. (2016). Sistema de análisis estático de vulnerabilidades para PHP. *In Décima Edición de La Semana Tecnológica*, (January 2011).

Tiirik, K. (2004). Comparison of SDL and Touchpoints. *Last Retrieved*, *11*, 1–18.

Verde, L., & López, N. (2013, October). EL CICLO DE DESARROLLO DE LOS CLUSTERS. *Congreso Internacional de Contaduría, Administración e Informática*, 1–13.

Von Solms, B. (2005). Information Security governance: COBIT or ISO 17799 or both? *Computers and Security*, *24*(2), 99–104.

Wichers, D. (2017). OWASP Top 10 - Los diez riesgos más críticos en Aplicaciones Web. *OWASP Foundation*.

Yourdon, E. (1975). *Techniques of program structure and design*. Prentice-Hall.

Zubrow, Da. (2004). *Measuring Software Product Quality : the ISO 25000 Series and CMMI*. CARNEGIE-MELLON UNIVERSITY PITTSBURGH PA SOFTWARE ENGINEERING INSTITUTE.