Sistema de Control de Versiones para el Desarrollo de Software Seguro

Director: Luis Eduardo Baquero Rey

Diana Marcela Trujillo Silva
Alfonso Chávez Baquero.

Fundación Universitaria los Libertadores.
Ingeniería de sistemas.
Pasantía Investigativa.
Agosto 2016

Hoja de Aceptación

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Firma del presidente del Jurado

_____

Firma del Jurado

_____

Firma del Jurado

Bogotá, agosto 2016

Dedicatoria

Dedico este proyecto de grado principalmente a Dios, por permitirme culminar esta meta a pesar de los obstáculos que se pudieron presentar, y por permitirme estar en un proceso de aprobación de proyecto.
A mis hermanos Viviana, Ricardo A., Felipe, Lina y demás familiares que creyeron en mí, en mi proyecto de vida, en las metas que me he propuesto como persona, mujer y próxima profesional.
Así mismo, a mis padres, Yolanda Silva y Ricardo Trujillo, quienes inculcaron desde siempre los principios que han hecho de mí y mis hermanos, personas de bien con el firme propósito de servir a la sociedad y cumplir los objetivos propuestos, quienes me han apoyado y acompañando en cada uno de los pasos que como profesional di en esta carrera y, al gran logro que estoy cumpliendo en el momento de obtener el título de Ingeniero(a) de Sistemas.

A mi compañero de proyecto Alfonso Chávez, por el gran esfuerzo y dedicación brindado en este semestre, en el que fueron sacrificados descansos, tiempo con nuestros seres queridos y que, hoy por fin, estamos viendo sus frutos.

Diana Marcela Trujillo

Dedico este proyecto de Grado principalmente a DIOS por haberme permitido llegar hasta este punto y haberme dado la salud, la sabiduría y el entendimiento para alcanzar mis objetivos.
Dedico este proyecto a DIOS por todos esos momentos en que le pedí fortaleza para no desvanecer, para continuar y no morir en el intento y por qué sé que siempre estará a mi lado guiándome por cada nuevo paso que dé.
A mi Hijo Daniel Felipe Chávez, por ser mi más grande inspiración y mi propósito de superación, siempre has estado en mi mente y mi corazón.
A mis padres por sus oraciones y por ser quienes a lo largo de mi vida han velado por mi bienestar y educación siendo mi apoyo en todo momento, depositando su entera confianza en cada reto que se me presentaba sin dudar ni un solo momento de mi inteligencia y capacidad.
A mi querida Yuly Flórez por estar ahí siempre motivándome con su incondicional apoyo, por su fe en mí y por estar presente en cada paso que doy.
A mis hermanos, cuñado, familiares cercanos y personas allegadas a quienes presento especial cariño y respeto, porque siempre creyeron en mí, me apoyaron y que gracias a ellos soy lo que soy ahora.
A Diana Marcela Trujillo por ser mi compañera de estudio y compañera en este Proyecto de Grado, por su profesionalidad, su entrega y dedicación para que este trabajo sea nuestra carta de presentación como los profesionales que ahora somos.
A los profesores de La Fundación Universitaria Los Libertadores que aportaron y dieron lo mejor de sus conocimientos en mi formación profesional, a todos ellos mi dedicatoria y agradecimientos.

Alfonso Chávez Baquero

Agradecimientos

Al Ingeniero Luis Eduardo Baquero Rey por su gran aporte, direccionamiento y guía en el desarrollo de este proyecto.
Al Ingeniero Miguel Hernández Bejarano, quien compartió con nosotros sus amplios conocimientos en procesos de seguridad y aplicación a metodologías de desarrollo de software.

A cada uno de los profesores e instructores de la Fundación Universitaria Los Libertadores, con los cuales, ya sea mediante clases, tutorías y/o acompañamiento, nos inculcaron en el transcurso de esta carrera el profesionalismo y pusieron a nuestra disposición cada gramo de su conocimiento.

Resumen

Existen metodologías creadas para llevar un Ciclo de vida en el Desarrollo de Software Seguro que pueden ser aplicadas en cualquier técnica definida en un proyecto de software, sin embargo, en la actualidad no existe un Sistema versionador de documentos que facilite el control de sus entregables; este documento, contiene información sobre la importancia del proceso de seguridad en el desarrollo de sistemas de Información, cuales son las principales técnicas en el mercado especializadas para llevar un ciclo de vida seguro (S-SDLC) y, propone el desarrollo de una metodología segura para el proceso.

Palabras claves:

Sistema Control de Versiones, Metodología para el desarrollo de Software Seguro, Seguridad en el software.

Tabla de Contenido

Lista de tablas

Lista de figuras

INTRODUCCIÓN

La aplicabilidad de métodos sistemáticos para implementar ciclos de vida para desarrollo de software seguro, apalanca los procesos de creación de software de calidad, dando al cliente la fiabilidad de contar con un Sistema de información seguro, que desde el primer instante del planteamiento ha contado con estrategias, procesos y prácticas que garantizaran el producto. El desarrollo de un repositorio versionador que cuente con cada paso, tarea o actividad que puede ser implementado en cada una de las fases del proyecto de desarrollo, facilita a los programadores herramientas que permitan minimizar las amenazas presentes en este tipo de producto al igual que, da un paso más en el proceso de identificación y mitigación de riesgos cada vez más comunes en la actualidad.

El cliente o programador debe tener el control de su proceso y las tareas que debe realizar, por lo que es factible que pueda estructurar incluso nuevas técnicas basadas en las metodologías o estrategias de diversos autores; esto permitirá, abarcar y cubrir las posibles falencias no identificadas hasta el momento acorde al status actual y los requerimientos de seguridad que se quieran implementar.

Con tal fin, este proyecto se enfoca desde el punto de vista de la seguridad en procesos de desarrollo de software, una guía que permita fortalecer y adoptar una orientación crítica respecto al mejoramiento de la calidad y la eliminación de vulnerabilidades frente a ataques que puedan poner en riesgo la integridad del Software desarrollado, mediante la construcción de un software versionador que facilite el seguimiento de las metodologías y fases de software seguro que se deseen implementar en los proyectos de software.

Capítulo 1

Descripción del Proyecto

1.1 Planteamiento del Problema

En la actualidad, el desarrollo de software se encuentra encaminado a la aplicación de metodologías que rigen su ciclo de vida (SDLC) y que son usados de manera específica adaptándose a circunstancias individuales de cada proyecto como tiempo, presupuesto, recursos (Ejemplos: cascada, iterativas, agiles, etc); existen así mismo técnicas perfeccionadas para que estos ciclos de vida sean desarrollados de manera segura, de tal manera, que las vulnerabilidades y riesgos que en algunas ocasiones se evidencian al culminar los proyectos, puedan ser previstas y mitigadas minimizando amenazas en el desarrollo, entrega y ejecución del producto al cliente.

Existen Sistemas Controladores de Versiones habitualmente encargados de llevar el control de cambios ejecutados por los programadores mientras trabajan en un código en común, facilitando la validación de versiones anteriores con el fin de trabajar sobre una copia del código y, garantizando la corrección de posibles errores sobre una fuente confiable, sin embargo, no existe en el momento un Sistema Versionador que permita a los desarrolladores y al administrador del proyecto, escoger una metodología enfocada a llevar un ciclo de vida para el desarrollo de software seguro o que, tenga la posibilidad de crear técnicas personalizadas con sus respectivos artefactos o documento.

Se han creado innumerables estadísticas respecto a la exposición, violación de datos y ataques informáticos en el mundo, sin embargo, vale la pena mencionar el artículo de la revista Dinero del 01/05/2016 donde se hace referencia al informe realizado por

Certicámara en el cual, se indica como un 25% de estos ataques se presentaron en Latino América; enfocado en Colombia, se destaca la aplicabilidad de nuevas herramientas tecnológicas como la biometría por huella y voz en entidades estatales. Tal como se evidencia en la Tabla 1, los ataques informáticos en Latinoamérica tenderán a aumentar:

Tabla 1. Cifras ataques informáticos en América Latina

Sectores	**Ataques por día**	**Porcentaje**	**Tendencia a futuro**
Financiero	6.600.000	75,29%	Aumentarán
Gobierno	925.600	10,56%	Aumentarán
Comunicaciones	737.200	8,41%	Se Mantendrán
Energía	325.347	3,71%	Descenderán
Industria	173.900	1,98%	Aumentarán
Comercio	3.600	0,05%	Aumentarán
Total	**8.765.647**	**100%**	

Así mismo, un estudio estadístico publicado por la revista Enter.co titulado "*El 68% de las empresas en Latinoamérica sufrió ataques informáticos"*, donde fueron dados a conocer los resultados del "Estudio Global sobre la Seguridad TI Empresarial", evidencia como fue abarcada información de 3.300 empresas distribuidos en 22 países diferentes. Las grandes empresas, son el objetivo principal de ataques como phishing y ciberespionaje, entre otros, por lo cual, el uso de mecanismos de prevención y fortalecimiento de la seguridad en el desarrollo de software, puede disminuir las vulnerabilidades a las que puedan verse enfrentadas; el estudio arrojó que, aunque un 25% de los profesionales de TI

están en capacidad de enfrentar estas amenazas, solo un 16% lo realiza proactivamente estos procedimientos.

Datos estadísticos de Colombia también han sido dados a conocer en el artículo: "*Colombia es el tercer país de la región con más ataques cibernéticos*" del periódico el Tiempo en septiembre del 2015, la información específica como para este año, Colombia representa un 21,73% dentro de los países latinoamericanos con mayor nivel de ataque informático, seguido por un 13,94% en Argentina y, por un 11,22% en Ecuador y Perú.

Lo anterior, evidencia la necesidad actual de contar con un Sistema de Control de Versiones, enfocada en los Artefactos definidos en las Metodologías de Desarrollo Seguro, que permita contar con un repositorio que facilite la gestión documental de los diferentes entregables de las fases, teniendo un seguimiento sistemático de los proceso seguros del desarrollo y que, con cada aceptación de entregable, pueda ser garantizado que se cumple con los requerimientos mínimos de seguridad en el desarrollo.

1.2 Justificación

Con el creciente avance de las tecnologías se hace patente la necesidad de implementar nuevos sistemas de organización y seguridad, cada vez más sofisticados y tendiente a la eliminación de los errores por percepción humana en el desarrollo de Proyectos de software. Con el surgimiento de las tecnologías digitales y el progreso en el procesamiento y análisis de las mismas, surge un nuevo paradigma de seguridad, encarnado en el reconocimiento de patrones como su instrumento de interacción con la realidad y la validación de los factores críticos.

El enfoque metodológico del desarrollo de software hacia la prevención y menos a la corrección de incidentes de seguridad, debe ser el punto clave del por qué en la actualidad, el inicio y construcción de proyectos de software deberá tender un nivel de garantía al usuario de que el producto entregado incluye no solo sus requerimientos, sino que está listo para soportar los ataques al que pueda enfrentarse.

El Sistema de Control de Versiones para el Desarrollo de Software Seguro cumple con este objetivo, debido a que hoy por hoy, no existe un sistema que apalanque el seguimiento de seguridad en el desarrollo, por lo que, apoyándose en él, el administrador del proyecto podrá tener control respecto a las revisiones, modificaciones y cambios en los documentos que aseguran y sustentan, los procesos de seguridad que se han implementado en cada proyecto de software que lidere y así mismo en cada una de sus etapas.

1.2.1 Técnica

El Sistema de Control de Versiones para el Desarrollo de Software Seguro, contribuye con la gestión documental de artefactos haciendo uso de una base de datos que,

lleva el registro, el repositorio y el seguimiento de las diversas modificaciones realizadas a los documentos. Permite la ejecución en inglés y español, modificable desde la opción inicial del Sistema, su funcionalidad es intuitiva para el usuario final aplicando usabilidad necesaria.

1.2.1 Organizacional

A nivel organizacional, se ha definido el uso del Sistema por diferentes empresas previamente creadas, a las cuales, se les ha dispuesto de perfiles de visualización, creación y modificación por roles que garantizan la Confidencialidad de la información.

1.3 Línea de Investigación: Ingeniería de Software

Siendo la Ingeniería de Sistemas la conceptualización y engranaje de métodos sistemáticos y herramientas encaminados al desarrollo de productos de Software, donde se establecen procesos basados en mejores prácticas e impartiendo calidad al proceso desarrollado, la Fundación Universitaria los Libertadores, ha establecido un grupo de Semilleros de Información basados en estos principios, con el fin de sembrar dentro de la comunidad educativa, actitudes de carácter investigativo que a mediano y largo plazo se constituirá como uno de los pilares de crecimiento del estudiante. Como línea de investigación dentro del programa de Ingeniería de Sistemas, se ha conformado el grupo de Seguridad Informática que a su vez, hace parte del grupo de Investigación y Desarrollo de Nuevas Tecnologías de la Información y Telecomunicación (GRIDNTIC (Colciencias.gov, 2009)) establecida a nivel nacional, en el cual, se hace latente el uso de la Ingeniera de Software en el desarrollo de los proyectos investigativos propuestos o en marcha, dando el sello del uso de buenas prácticas.

1.4 Delimitaciones

Entendiendo la aplicabilidad de la Ingeniería de Software en el desarrollo de proyectos y la Línea de Investigación, los objetivos y fines con los cuales se usa, el proceso ejecutado para el Sistema de Control de Versiones para el Desarrollo de Software Seguro, ha contado con el análisis económico, de recursos humanos y técnicos, que guio las diferentes etapas de la metodología que se implementó, para cumplir con cada uno de ellos, entregando al usuario un Producto de Calidad, que cumple con los requerimientos iniciales que son recolectados en el proceso de análisis y evaluación del entorno.

Con el fin de establecer las limitaciones y alcances del Sistema de Control de Versiones para el Desarrollo de Software Seguro, ha sido evaluada la Viabilidad y Factibilidad del producto desde el punto de vista del Cliente como la Fundación Universitaria los Libertadores.

1.4.1 Factibilidad Económica

En la revisión de equipos realizado al cliente, y los encontrados en La Fundación Universitaria los Libertadores, garantizan el correcto funcionamiento del desarrollo, por lo cual, se convierte en un proyecto factible económicamente para el cliente. De acuerdo a los avances en TI, es susceptible a cambios de los recursos técnicos y lógicos con el fin de soportar la ejecución del proyecto metodológico en la implementación de nuevas versiones.

1.4.2 Factibilidad Técnica

Teniendo en cuenta que el sistema control de versiones será realizado vía WEB, los equipos con los cuales se sugiere ser ejecutado deberán cumplir con los siguientes requisitos:

- Sistema Operativo 64 bits, para equipos Microsoft - Windows 7 o versiones superiores.
- Tarjeta de red acorde al último avance tecnológico en el momento de ser realizado el análisis.
- Conexión a internet mediante cable o conexión inalámbrica.
- Hosting
    - Capacidad: 500 MB o superior
    - Sistema Operativo: Linux o Windows
- Dominio:
    - Protección activa
    - Tiempo: 1 año
- Firewall – Proxy y Gateway de aplicaciones, que permita encontrar y analizar el contenido para que no infrinja la seguridad del sistema.

1.4.3 Factibilidad Operacional

Al tratarse de un Servicio WEB y teniendo en cuenta que la Fundación Universitaria Los Libertadores maneja distintas sedes educativas, el producto de software podrá ser operado de la siguiente manera:

- Usuario destinado para el uso, podrá ingresar a su computador ya sea Personal o Desktop, mediante conexión alámbrica a internet, podrá ingresar con su usuario y contraseña, a la base de datos y ejecutar de esta manera el proceso específico, tal como se evidencia en la Figura 1:

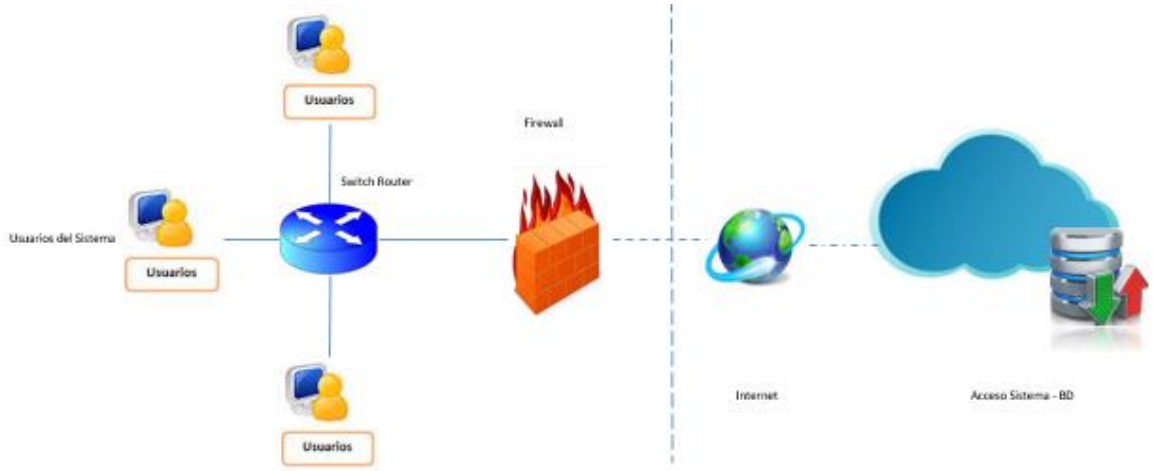


Figura 1. Conexión alámbrica.

- Usuario destinado para el uso, podrá ingresar a su computador a sea Personal o Desktop, mediante conexión inalámbrica a internet, podrá ingresar con su usuario y contraseña, a la base de datos y ejecutar de esta manera el proceso específico como se muestra en la figura 2

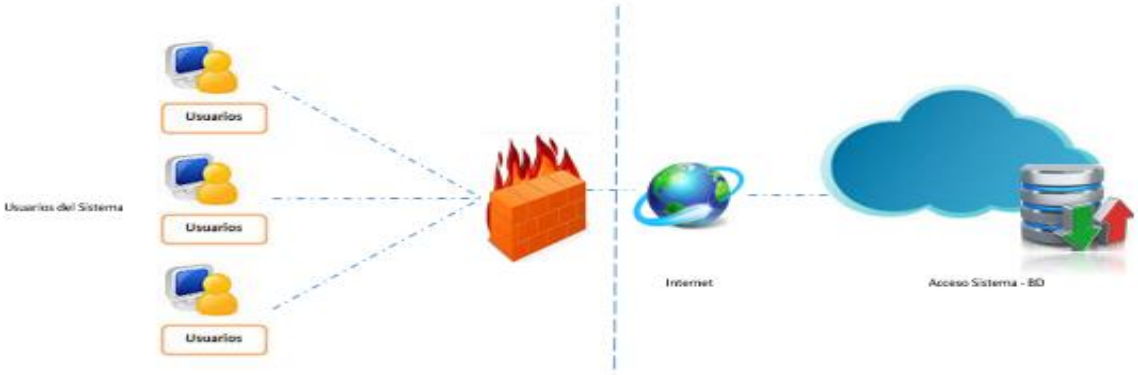


Figura 2. Conexión inalámbrica.

De esta manera se garantiza que el sistema es factible operacionalmente al permitir la conexión con diferentes medios, diferentes equipos y diferente ubicación geográfica.

1.4.4 Viabilidad Económica

Los equipos de cómputo con los que cuentan los usuarios de la Fundación Universitaria Los Libertadores, cumplen con las factibilidades económicas, técnicas y operativas para garantizar el uso del sistema de información, lo que impulsa una viabilidad económica al no incurrir en costos adicionales relacionados a mejoramiento de la arquitectura, incluso, abarcando las sucursales fuera de Bogotá.

1.4.5 Viabilidad técnica

Acorde a las necesidades del cliente, podría ser necesario el cambio o mejoramiento de los equipos, en caso de ser requerida una nueva versión del Sistema de Información. Estos cambios en el modelo se encuentran fuera del alcance de este proyecto; por lo cual, se hace claridad en que, aunque en el momento el proyecto es viable técnicamente, puede ser sugerido cambio en la estructura actual después del análisis de las modificaciones que generen la nueva versión.

Adicional a estas validaciones, el servicio suministrado por el producto de software no presenta limitaciones territoriales, esto gracias a la posibilidad de cambiar de idioma para el ingreso. Así mismo, se plantea el uso para diferentes empresas desarrolladoras de Software con garantía de calidad y seguridad de sus documentos e información sensible.

1.4.6 Cronología

En el planteamiento del plan de trabajo para el desarrollo del Proyecto, han sido identificadas actividades relacionadas a cada una de las fases que aplican en la Metodología de desarrollo de software elegido para este fin. Para tener su detalle ver Anexo A – Cronograma de Actividades.

1.4.7 Metodología

Para el desarrollo de este proyecto, se manejaron dos propuestas dentro de la parte investigativa, una preliminar donde se llevó a cabo el proceso de levantamiento de información donde inicia el proceso investigativo a nivel teórico, conceptual y legal que será agrupado en un marco referencial, y que son la base que determina la solidez para abordar el problema y, por otro lado, el uso de una metodología ágil en este caso Xtreme Programming (XP), en la cual el foco de la investigación se inicia en la fase de planeación donde evalúa el manejo actual y la obtención de información operacional básica para iniciar con el diseño del producto de Software.

Basando su procedimiento en la relación e interrelación de los desarrolladores y el cliente, XP ayuda de manea eficaz a cumplir satisfactoriamente con las expectativas planteadas, brindando adicionalmente calidad al software entregado; teniendo en cuenta que las fases del Ciclo de Vida de Desarrollo de Software pueden ser ejecutadas, poco a poco, se garantiza de alguna manera el cubrimiento de las modificaciones que sean necesarias en el camino; Estas fases se encuentran relacionadas en la Figura 3.

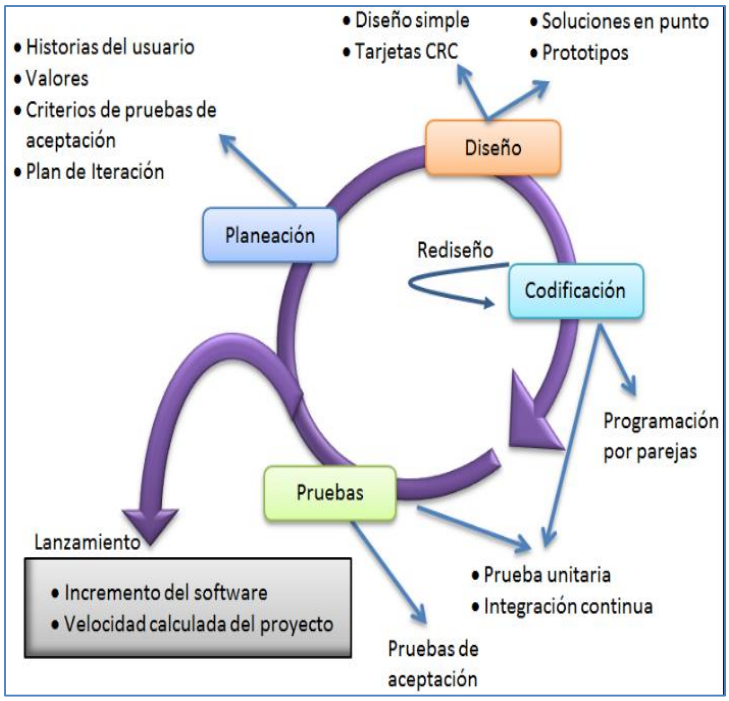


Figura 3. Fases Metodología XP (http://slideplayer.es/slide/2273638/)

1.4.8 Financiera

Teniendo en cuenta el desarrollo del Producto de software que será creado bajo software libre, el presupuesto estimado se detalla en la Tabla 2.

Tabla 2. Presupuesto financiero

Item	***Descripción***	***Valor***
Computadores	Lenovo: Intel Core7, HD 1TB, 4GB RAM, 2,30 GHz HP: AMD A8 PRO – 7150B R5, 10 Compute Cores 4C+6G 1,90 GHz, 8GB RAM Table data	$ 3’150.000
Desarrollo del proyecto	Tiempo de trabajo de los integrantes por 7 meses	$ 18’500.000
Hosting	Por 1 año Capacidad: 500 MB Sistema Operativo: Linux	$ 75.000
Papelería	Impresiones, Carpetas, CD	$ 390.000
Dominio	Por 1 año Protección activa	$ 44.990
Servicio de Internet Table data	7 Meses de desarrollo	$ 300.000
Total		$22’459.990

1.5 Objetivos

1.5.1 General

Diseñar, desarrollar e implementar un Sistema de Control de versiones para entregables aplicados en Proyectos de Software usando Metodologías del Desarrollo de Software Seguro.

1.5.2 Específicos

- Diseñar un repositorio que permita al usuario iniciar un proceso de control de seguridad en el desarrollo de software.
- Permitir el acceso a las versiones de artefactos en cada fase del proceso.
- Facilitar el proceso de seguimiento online para programación de artefactos o entregables, mediante la inclusión de estados de revisión, garantizando la calidad de la documentación del Sistema.

Capítulo 2

Marco Referencial

2.1 Marco Teórico

Teniendo en cuenta que el punto central de este proyecto es crear un producto de software que pueda llevar las versiones respecto a los documentos para el desarrollo de software seguro, será indispensable para su continuidad, abordar algunos parámetros que servirán de base conceptual para la evolución del mismo. Como inicio de la investigación es primordial contar con la claridad para definir las causas por las cuales en la actualidad hablar de software seguro se ha convertido en un hito para los procesos de desarrollo, de que se trata, que metodologías existen y como deben ser aplicadas.

Aplicar procedimientos para fortalecer y aplicar protocolos de seguridad al software, más que en un valor agregado, se convierte en el sello de calidad que cualquier cliente espera recibir de un producto, la vulnerabilidad a ataques se convierte en un riesgo latente que debe ser mitigado o prevenido a la brevedad desde el mismo momento de la concepción del proyecto.

En la actualidad, diferentes organizaciones han expresado su preocupación referente a la seguridad y al desarrollo de estándares o técnicas que permitan conceptuar un ciclo de vida seguro para el desarrollo de software puesto que, la seguridad se ha convertido en el principal aliado de los proceso de desarrollo de software y ha pasado de ser uno de los puntos a tener en cuenta en los requerimientos funcionales y no funcionales, a la creación y aplicación de metodologías encaminadas al establecimiento de un ambiente libre de debilidades a los ataques, que evolucionan a pasos agigantados.

2.1.1 Sistema Control de Versiones

Un Sistema Control de Versiones puede definirse como un Software encargado de llevar el control de las diferentes versiones o modificaciones realizadas en el desarrollo de software, en este, es posible administrar y llevar el registro sistemático de los avances que se puedan tener en el mismo.

Generalmente este tipo de software es utilizado en el proceso de gestión de código dado al trabajo de varios desarrolladores en un mismo producto, conservando una copia de cada uno de los cambios realizados y garantizando de esta manera el trabajo integrado del equipo del proyecto. Dentro de las ventajas encontradas en el uso de estos sistemas tenemos:

- Reconstrucción de archivos o versiones anteriores cuando se requieran.
- Bitácora de modificaciones y cambios.
- Registro de todo tipo de gestión en el código fuente.
- Recuperación de versiones antiguas desarmando de alguna manera los cambios realizados en la última entrega o modificación.

Se encuentran divididos en dos grupos:

2.1.1.1 Sistema Control de Versiones Centralizados

Como su nombre lo indica, este tipo de Sistema de control almacena la información en un servidor centralizado donde se encuentra el proyecto y cada una de sus modificaciones, los diferentes desarrolladores deberán entrar al servidor, descargar una copia en su equipo, realizar las actualizaciones que hubiera lugar y, subir esta nueva versión para llevar el control de los procesos. Posibles desventajas:

- Conexión continua a una red para tener acceso al servidor central.
- Posibles retrasos en los tiempos en caso de presentarse caída de red o del servidor.

Dentro de este tipo de Sistemas Control de Versiones se encuentran:

- CVS (Concurrent Version Systems): Favorece y facilita el trabajo colaborativo entre diferentes desarrolladores mediante la arquitectura Cliente-Servidor. Son usados principalmente en el control de cambios al código fuente de un proyecto, sin embargo, su uso incluye al versionador de documentos de toda índole que puedan tener diferentes modificaciones. En cuanto a su funcionamiento, en caso de concurrir dos desarrolladores en el trabajo de un mismo código, el CVS permite el proceso de actualización generando una nueva versión del código fuente, esto, en caso de no presentar conflicto al intentar tocar una misma línea por ambos actores, si este es el caso, se indicará al usuario y no será generada la nueva versión. Adicional, el sistema ofrece un proceso de log de proceso en el cual se registrarán los datos básicos de la modificación realizada como fechas y usuarios usados.
- Subvesion: Sistema Control de Versiones de código abierto cuya utilización se extiende desde documentos hasta estructura y diferentes versiones de directorios, permitiendo su repositorio. Este sistema facilita el proceso de trabajo en paralelo de diferentes desarrolladores, cuya actividad final será la integración conformando una última versión usando el proceso "Copiar – Modificar – Unificar", de esta manera, evita el bloqueo de usuarios al intentar ejecutar cambios en la misma línea de código. Cada integrante deberá crear una copia y de esta manera, trabajará en un

documento personalizado que por medio de “Subversión” será compactada en la versión final.

2.1.1.2 Sistema Control de Versiones Distribuidos

Permite que cada integrante pueda realizar el proceso de manera local e independiente para ello es necesario clonar el repositorio del proyecto en el equipo local, de esta manera, se permite la generación de versiones independientes en cada desarrollador y que, cuando sea definido podrá sincronizar su información en el servidor.

Funcionalmente permite la fragmentación en ramas de las diferentes actividades y que, pueden facilitar el avance de diferentes avances del proyecto.

- No es necesario trabajar por red, ya que cada uno podrá trabajar en una copia local.
- Permiten el envío de versiones de un desarrollador a otro sin necesidad de pasar por el servidor.
- Uno de los desarrolladores será el encargado de recopilar la información y enviarla al repositorio remoto encontrado en el servidor.

Entre los más usados se encuentran:

- Sistema Control de Versiones GIT: Desarrollado por Linux, funcionalmente, el desarrollador deberá descargar una versión local del proyecto, en la cual realizará las modificaciones que hubiera lugar, posteriormente Git generará un listado de cada uno de los archivos modificados, los cuales eran seleccionados para crear la nueva versión; Después de ser confirmados los cambios que se han almacenados en el “área de preparación” y en el Directorio Git, generando de esta manera una modificación en su repositorio local. Al ser trabajado paralelamente por varios

desarrolladores, Git realiza la función Merge para compactar las diferentes versiones creadas por cada uno, esto teniendo en cuenta que al usar este sistema es creada una rama maestra de la cual se desprenderán muchas otras ramas que tendrán incluidas todas las historias del proyecto.

Una rama es una abstracción que permite trabajar de forma paralela sobre un mismo proyecto, esto sin afectar el resto de proyecto.

Está compuesta por 3 unidades:

- Directorio de Git: Repositorio de objetos en los cuales se ha desarrollado modificación, contiene así mismo el historial de cambios.
- Directorio de trabajo: Almacena los archivos iniciales sobre los cuales se realizarán los cambios.
- Área de preparación: Contiene la información de los archivos modificados que serán enviados al ser confirmados por el desarrollador.

- Mercurial: Con funcionamiento similar al Git, este Sistema Control de Versiones, guarda una copia de los archivos e historial del proyecto de manera local, pero, no se conecta directamente con el Repositorio Origen, este proceso debe ser llevado a cabo por el desarrollador, sin embargo, cuenta con un aplicativo WEB que facilita las siguientes funciones:
  - Navegación de la composición estructural del proyecto
  - Visualización del antecedentes y cambios
  - Expansión de archivos y directorios

- Permite el ingreso de usuarios remotos para ejecutar actividades de modificación, copia y actualización de los repositorios.

2.1.2 Secure Software Development Life Cycle (S-SDLC)

El ciclo de vida de desarrollo de software seguro establece las fases o pasos que un software debe seguir con el fin de fortalecerse, cumpliendo a su vez, con los requerimientos del usuario final y permitiendo la generación de software de calidad.

Intentar proveer al cliente un sistema seguro en un 100% puede constituir una meta irreal y un tanto utópica, teniendo en cuenta que a media que estas técnicas para el fortalecimiento han evolucionado, la tecnología así mismo, ha proveído un desarrollo y entendimiento más amplio a personas inescrupulosas para el fortalecimiento de procedimientos que pueden perjudicar el software final.

2.1.3 Objetivos de la seguridad

Los objetivos de la seguridad en el desarrollo de software planteado por John Viega y Gary McGraw (Mc Graw & Jhon, 2006), deben tener como punto de partida la respuesta a la pregunta "¿Seguro contra qué o quién? Así pues, aparecen los tenemos los siguientes objetivos que se deben garantizar con este proceso:

***2.1.2.1 Prevención*:** Tal como se presenta en todas las ramas y dependencias de la seguridad tanto a manera de Sistemas de Información como general, la prevención es una de las actividades con menos énfasis en el proceso de proyectos, por lo cual, muchos de los ataques y perjuicios a los desarrollos, se presentan debido a las grietas ocasionadas por las apariciones de vulnerabilidades que no fueron atendidas en su momento o con anterioridad a su presencia.

Las aplicaciones online con ejecución en Internet, constituyen uno de los puntos débiles en el desarrollo de software seguro por lo que llevar un control de riesgos presenta una complejidad mayor y, los golpes que llegan a ser efectivos se pueden propagar fácilmente. La identificación, seguimiento, mitigación y control de los riesgos, impulsa el reforzamiento de la seguridad desde el punto de vista de prevención.

***2.1.2.2 Trazabilidad y Auditoria*:** Un objetivo primordial de la seguridad del software es llevar la trazabilidad de los procesos con el fin de conocer de qué manera, cuando, quien, y como, ha sido vulnerado el software. El proceso de auditoría de sistemas, debe hacer parte de la estructura organizacional de cada compañía, en la que se busca validar que las normas internas y externas establecidas se cumplan, en este caso, con la privacidad de la información y todos aquellos mecanismos que faciliten el blindaje del desarrollo, haciendo las labores de un Control Interno.

***2.1.2.3 Autenticación*:** En la actualidad, el mecanismo establecido de asignación de contraseña a cada usuario, establece el nivel de autenticación que les permite, ingresar, visualizar, modificar, ya sea desde las vistas o, directamente en las bases de almacenamiento. La asignación de perfiles permite controlar los accesos y de esta manera blinda los datos de la manipulación indeseada por usuarios sin autorización.

En el ámbito específico de Internet, la mayoría de los usuarios identifican el icono de un “candado” como la comprobación de ingreso a una página segura (SSL – Secure Socket layer) sin embargo, más allá de esto, la seguridad y autenticación de usuarios se realiza mediante el uso de criptografía. La autenticación es crítica en el desarrollo de software seguro.

***2.1.2.4 Integridad*:** La integridad hace referencia al hecho de identificar si algo fue modificado desde su creación. A diario, las personas usan las conexiones a internet como WAP (Wireless Application Protocol) como medio para la transmisión de información. Mediante este objetivo, se garantiza que los datos como por ejemplo los valores de la bolsa, no sean modificados; a nivel general, se ha identificado que las usurpaciones en transacciones electrónicas son más sencillas que incluso falsificar dinero.

2.1.4 Objetivos de un proyecto de software

Así como los anteriores puntos nos definen los objetivos principales en el desarrollo de software seguro, es de primordial importancia abordar los objetivos de un proyecto de software. Al alcanzarlos, el producto a entregar cumplirá con los propósitos fundamentales de un Sistema de Información. A continuación, se especifican los 5 principales objetivos:

2.1.3.1 Funcionalidad

Es el pilar de los objetivos establecidos en este tipo de proyectos, la funcionalidad hace referencia a la capacidad que debe tener el desarrollo de cumplir a cabalidad con los requerimientos funcionales planteados por los usuarios. Esta definición de requerimientos debe ser planteada en las primeras fases de ciclo de vida con el fin de identificar lo que se espera como resultado y así mismo debe, especificar claramente y delimitar con exactitud la funcionalidad esperada. Generalmente se establecen como Historias de usuarios o los Requerimientos Funcionales del usuario.

2.1.3.2 Usabilidad

La usabilidad se ha convertido en eje principal para un usuario final, en este es definida la facilidad y practicidad con la cual pueda ser manipulado el software, partiendo

de lo amigable que sea el desarrollo, mayor acogida tendrá entre los posibles compradores u clientes, sobre todo, en cuanto a aplicaciones web y a transacciones o procesos.

Mediante el uso de mecanismos criptográficos, puede generarse un ambiente cómodo para el usuario y, así mismo, brindar un ámbito de seguridad en procedimientos financieros como los ofrecidos por algunas entidades bancarias.

2.1.3.3 Eficiencia

Busca suministrar al usuario el rendimiento y la eficiencia que espera encontrar en un desarrollo, en la actualidad, debido al ritmo cambiante y extenuante de las actividades humanas, impacta considerablemente en la expectativa que se prevé frente a un producto de software; sin embargo, entendiendo la finalidad de crear Software seguro, es necesario adoptar procesos de autenticaciones que muchas veces pueden tomar algún tiempo pero que, a mediano y largo plazo pueden constituir barreras para el ingreso de intrusos al sistema.

2.1.3.4 Time to market

Este objetivo hace referencia al tiempo en que el producto alcanza su venta o entrega teniendo en cuenta el momento en que fue iniciado, para ello, se debe tener presente dos premisas claves al momento de cumplirlo:

- Esperar la maduración completa del producto, con el fin de cumplir con todas y cada una de las expectativas planteadas, mostrando un resultado completo que pueda superar la posible competencia o,
- Reducir las funcionales hasta llegar a una versión que pueda satisfacer las necesidades básicas para competir en el medio con los demás.

La agilidad en su desarrollo no es sinónimo de inseguridad, actualmente, la construcción de software puede cumplir con imprevistos relacionados a la lentitud y así mismo con las expectativas de seguridad necesarias.

2.1.3.5 Simplicidad

El desarrollo de un sistema simple, pero a la vez seguro, es una de las principales metas en cualquier proyecto de desarrollo.

2.2 Metodologías y Estándares de control para el Desarrollo de Software Seguro

Diferentes metodologías y estándares se han desarrollado con el fin de proveer un ambiente seguro en el desarrollo mediante la implementación de secuencia o pasos, y con ello adquirir la cualidad de resistir a posibles ataques. Entre ellas tenemos:

2.2.1 Correstness by Construction (CbyC)

Esta metodología busca la creación de código de manera correcta desde su inicio, procurando la corrección y eliminación de errores desde su ingreso, para ellos se apalanca en procesos rigurosos de seguridad, donde sus requerimientos presentan un alto detalle.
En una primera entrega, la metodología CbyC presenta una visión completa del sistema con funcionalidad en general limitada pero que permitirá los ajustes y mejoras en cada iteración (IEEE, 2002).

La secuencia establecida para la construcción de sus fases establecida en la Figura 4, tiene pautas encontradas en el desarrollo ágil, en el cual, se recibe retroalimentaciones a media que se entrega y valora el producto que está en construcción.

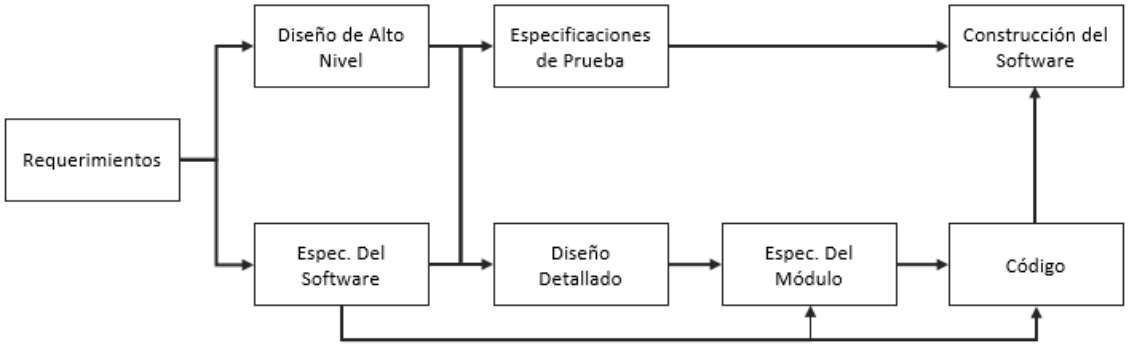


Figura 4. Proceso de fases para CbyC
(http://recibe.cucei.udg.mx/revista/es/vol2-no3/computacion05.html)

2.2.1.1Fase de requerimientos

Esta fase define los requerimientos de usuario, especificando las funciones y propósito, así mismo, tiene en cuenta la definición de requerimientos no funcionales expresados en los diagramas de clases. Cada requerimiento debe pasar por el proceso de validación de seguridad a la amenaza correspondiente.

2.2.1.2 Fase de diseño de alto nivel

Describe la composición gran escala del desarrollo incluyendo las bases de datos y funcionalidad, teniendo como punto principal los requerimientos no funcionales en cuanto a su ámbito de seguridad en cada punto álgido.

2.2.1.3 Fase de especificación del software

Tiene como finalidad especificar la interfaz de usuario (look and feel), en esta fase se desarrolla un prototipo para la posterior validación con el cliente.

2.2.1.4 Fase de diseño detallado

Define el conjunto de módulos, procesos y funcionalidades.

2.2.1.5 Fase de especificación de módulos

Se define el estado y comportamiento de los módulos para definir y garantizar el flujo de información, de esta manera, los efectos sobre incidentes seria mínimo.

2.2.1.6 Fase codificación

En esta fase de induce el desarrollo de pruebas con el fin de disminuir y eliminar los errores presentes en el código, de esta manera podrá ser definido en paralelo si debe ser creada una nueva versión del código. Con el fin de evitar ambigüedades, interviene en el proceso Spark.

2.2.1.7 Fase de especificaciones de pruebas

Como particularidad en la metodología CbyC, no es necesario ejecutar proceso de pruebas de unidad ni caja blanca, puesto que su foco central es realizar estas validaciones a nivel de sistema y con miras al cumplimiento de las especificaciones, comportamientos y requerimientos no funcionales.

2.2.1.8 Fase de construcción de software

La metodología presenta enfoque técnico en el cual, trata de disminuir los defectos, aumentando considerablemente su capacidad para minimizar fallas, lo que beneficia su implementación en el desarrollo de software.

2.2.2 Security Development Lifecycle (SDL)

Metodología compuesta por 16 actividades divididas en las fases que componen la técnica y que, están encaminadas en el mejoramiento del desarrollo de software; fue propuesto en 2004 por Microsoft e incluye un proceso de modelado de amenazas en el cual se busca identificar vulnerabilidades a nivel de código (Microsoft, 2016). SDL cuenta con

dos versiones de ejecución con el fin de tener un mejor acoplamiento dependiendo del tipo de desarrollo, ellas son:

SDL – Versión rígida: Enfocada en equipos de proyectos y desarrollo de productos de gran envergadura donde los cambios son mínimos.

SDL – Versión Ágil: Sus desarrollos son incrementales con aumento en la frecuencia de ejecución y seguimiento de actividades haciendo énfasis en su seguridad. Recomendada para desarrollos WEB.

Como parte diferencial de las metodologías de desarrollo de software seguro, dentro de sus fases, SDL cuenta con una etapa de "entendimiento de seguridad" y "aseguramiento" en al cual, se da seguimiento a cada uno de los resultados del proceso. A continuación, en la Figura 5, se indica el proceso de ejecución de cada fase planteada:

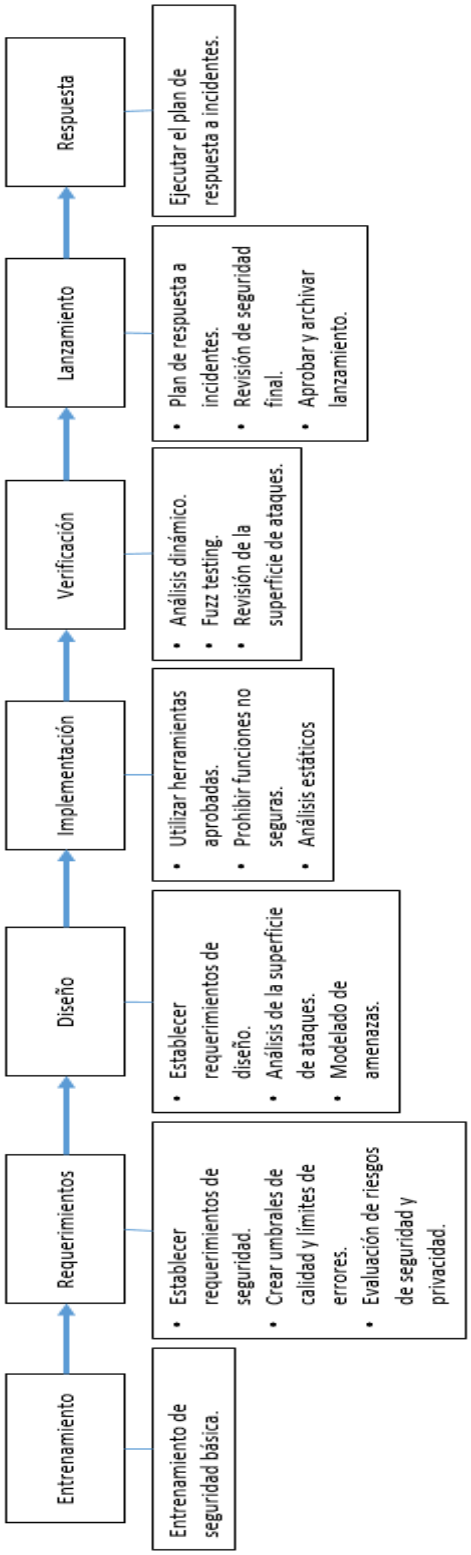


Figura 5. Fases y actividades metodología SDL (Microsoft Corporation 2007)

2.2.2.1 Fase de Entrenamiento

Esta fase contempla el proceso de capacitación al equipo de proyectos principalmente el grupo técnico, conformado con los desarrolladores, grupo de pruebas (tester) y administradores del sistema, con el fin de estar al tanto en las últimas actualizaciones en materia de seguridad. Esta actividad debe ser planteada mínimo una vez al año, de tal manera que el software pueda cumplir con requerimientos de seguridad frente a los ataques que pueda enfrentar y las nuevas amenazas que se puedan presentar.

Los temas básicos son

- Criptografía.
- Identificación y validación de riesgos.
- Técnicas y actualizaciones de seguridad.
- Inyección de SQL.

2.2.2.2 Fase de Requerimientos

En conjunto con un consultor de seguridad, debe ser revisado y planteado un procedimiento que incorporar las actividades de seguridad, para que, de esta manera, se cumpla con los requerimientos establecidos en las definiciones basadas en los requerimientos del cliente. Deben definirse las metas de seguridades que se propondrán para iniciar con el proyecto y continuar con una siguiente tapa de diseño, acorde a lo solicitado.

2.2.2.3 Fase de Diseño

Esta fase es la encargada de desarrollar el proceso de modelado de amenazas en cada componente, esto con el fin de identificar la posible aparición de riesgos y así mismo, implantar planes de mitigación, al igual que de identificación.

Son detallados los requerimientos y estructura con la cual se construirá la arquitectura del Software de manera segura, sin dejar aún lado los requerimientos iniciales planteados.

2.2.2.4 Fase Implementación

Teniendo en cuenta la ejecución y resultado de la fase de diseño en cuanto al modelaje de amenazas, el grupo desarrollador construirá y dirigirá su código de manera que puedan proteger el proceso y disminuir la afectación que puedan tener estas amenazas o ataques, para ello, se sugiere la codificación mediante el uso de estándares. En esta etapa se sugiere por Microsoft, el empleo de una herramienta de escaneo para el análisis de código estático y el uso de técnicas Fuzz Testing, respecto a las primeras, en el mercado existen diferentes tipos de herramientas dependiendo del lenguaje de programación usado, sin embargo, BugScout, Kiuwan, JustCode entre otras, hacen parte del grupo multilenguaje, y el uso de técnicas fuzz testing (Fernado Catoira, 2016), sugiere pruebas de ingreso de datos incorrectos o inesperados por el sistema para determinar el nivel de seguridad respecto a la validación de datos, la herramienta más usada para este fin es Fuzzer.

2.2.2.5 Fase de Verificación

En esta fase, en la que el software ya cuenta con una versión inicial y en la cual se desarrollan pruebas de validación de seguridad, se ejecutan revisiones íntegras y profundas en las secciones que se han identificado como blanco de ataques.

2.2.2.6 Fase de Lanzamiento

Como parte del proceso, esta fase debe ser ejecutada de dos a seis meses antes de la entrega del producto final al cliente, este proceso garantiza la tipificación de posibles vulnerabilidades nuevas o que han pasado desapercibidas en los pasos anteriores y que, guían el proceso hacia una fase encargada de afrontar dichas identificaciones. Su objetivo garantizar el aguante del software frente a ataques que se puedan presentar después de ser entregado al cliente y estar en una fase productiva.

2.2.2.7 Fase de Respuesta

Como valor agregado a la metodología, se pretende ampliar la cobertura en cuando a conocimiento de incidentes de seguridad, con esto, se garantiza no solo el control de los mismos si no, el desarrollo de actividades que puedan servir como previsoras en futuros desarrollos.

Esta metodología ha sido implementada por Microsoft como abanderada en los procesos de desarrollo de los diferentes Sistemas Operativos. Las experiencias adquiridas en los ataques de seguridad que han enfrentado y que posiblemente a afectado el producto, han sido fundamentales y hacen parte de la base de datos de conocimiento o errores con el cual se trabaja a diario, su fin es, desarrollar S.O. cada vez más estables y fuertes, con menos expectativa, mayor seguridad y calidad de su producto.

2.2.3 Common Criteria (ISO/IEC 15408)

Estándar internacional que permite a los desarrolladores definir las propiedades de seguridad y así mismo, validar que estas especificaciones se cumplan, para ello, se identifica Objetivo de Evaluación o sus siglas en ingles TOE - Target of Evaluation (Common Criteria, 2012) .

Un TOE puede ser definido como un Software, hardware, un conjunto de productos informáticos o parte de ellos; entre los inputs para el desarrollo de esta evaluación se tienen los documentos de diseño o resultados de pruebas de desarrollo ya ejecutadas.

Como ejemplo de un producto informático, se puede hablar de un Sistema Operativo el cual puede configurarse de diversas maneras como: tipos de usuarios, cantidad de usuarios, tipo de conexiones, etc.

Existen tres tipos de interesados en los procesos ejecutados en TOE:

- Consumidores o clientes: Son el foco y la razón de ejecutar procesos de evaluación, puesto que, se realizan con el fin de validar que los requerimientos iniciales se estén cumpliendo en manera de seguridad, así mismo, estos fueron clasificados mediante los análisis de riesgos. Esta metodología facilita una estructura denominada Protección de Perfil (PP), en el cual, es posible identificar los requerimientos sin que se presenten ambigüedades.
- Desarrolladores: Guía los desarrolladores en la preparación de Objetivos de Evaluación basado en los requerimientos de seguridad contenidos en una Declaración de Seguridad (Security Target – ST), este documento puede contener uno o más estructuras de PP identificados por los clientes o usuarios.

Evaluadores: Common Criteria estructura las actividades que serán ejecutadas en la fase de evaluación de los objetivos identificados.

En caso de encontrar fallas en los TOE, es importante:

- Probar detalladamente la TOE
- Validar los diseños del objetivo de evaluación
- Examinar los ambientes de seguridad física del TOE establecido.

Al identificarse la definición de un TOE, debe ser esclarecido cuando será usado una Declaración de Seguridad (ST), por lo cual, se determinan dos ítems de validación:

- Antes y durante la evaluación: Establece la relación primordial entre el desarrollador y el evaluador, puesto que, en esta Declaración de Seguridad, se define el "Que será evaluado".
- Después de la Evaluación: Se revisa "lo que fue evaluado", en este punto, la interacción clave se presenta entre el desarrollador y el usuario o cliente.

2.2.4 Systems Security Engineering Capability Maturity Model - SSE-CMM (ISO/IEC 21827)

Este estándar caracterizado por ser una métrica no basada en seguimiento de procesos, se encuentra plasmada en ISO/IEC 21827 Information technology -- Security techniques -- Systems Security Engineering -- Capability Maturity Model® (SSE-CMM®) (ISO, 2008), que tiene como alcance:

- Ciclo de vida de desarrollo general, incluyendo desarrollo, operación, mantenimiento etc.
- Funcionalidad integra de relación entre ámbitos del proyecto, teniendo en cuenta hardware, software, recurso humano, ingeniería de pruebas.
- Comunicación con otras entidades organizativas, gestiones del sistema, certificaciones, acreditaciones, entre otros.

2.2.5 Comprehensive, Lightweight Application Security Process (CLASP)

El uso de esta metodología, dirige a los desarrolladores en el proceso de revisión y validación desde las primeras etapas del ciclo de vida del desarrollo de software, de manera que este proceso se convierta en estructurado (OWASP, 2016).

Facilita su integración a cualquier desarrollo de software, articulando cada una de las actividades aplicados a uno o más roles previamente identificados como:

- Gerente del proyecto
- Arquitecto
- Especificador de requerimientos
- Diseñador
- Implementador (equipos de desarrollo)
- Tester (grupo de pruebas)
- Auditor de seguridad

2.2.5.1 Vista CLASP

La metodología plantea 104 fallas de seguridad agrupadas en 5 niveles de vistas, que tienen consigo actividades relacionadas en la figura 6:

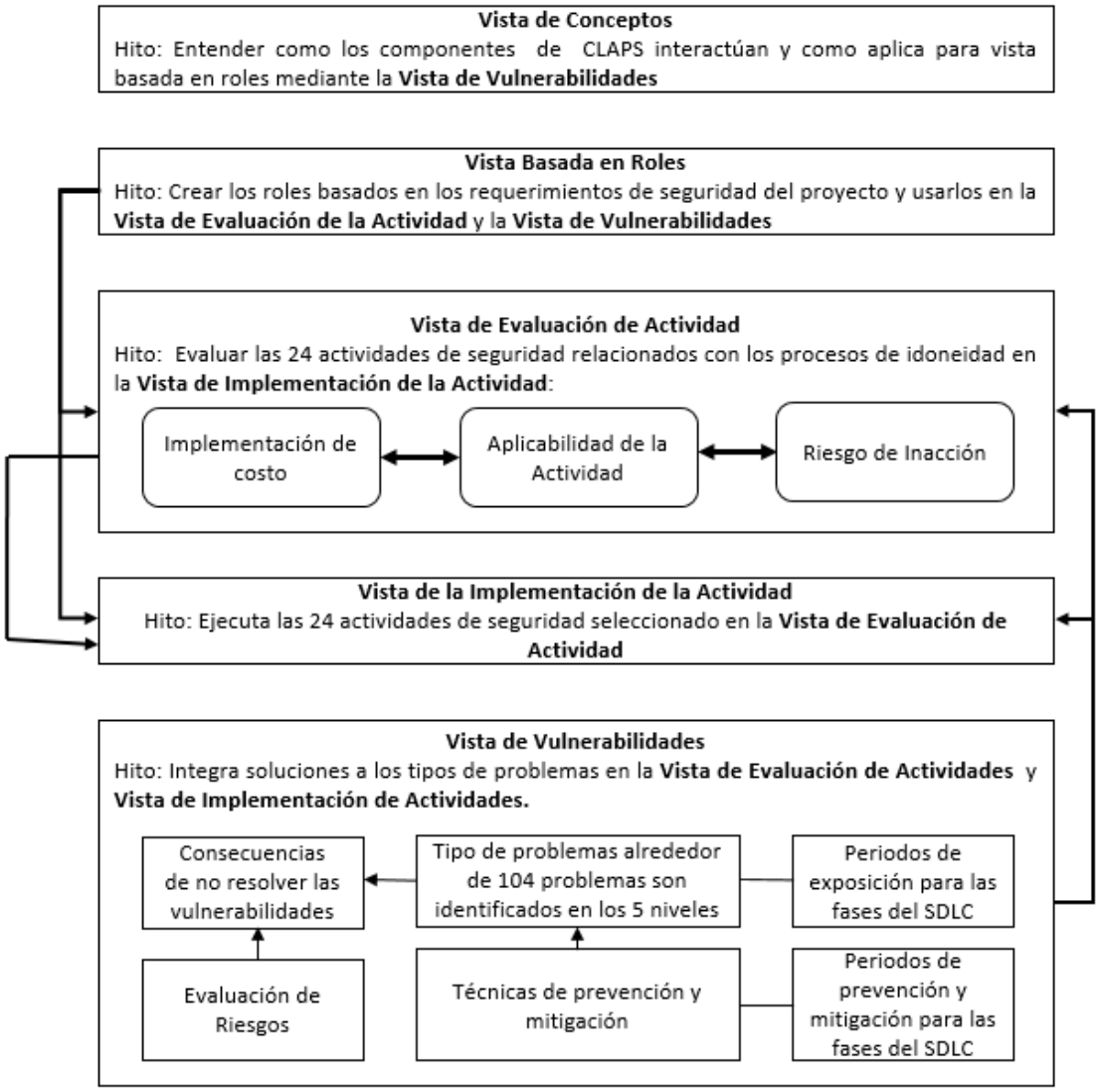


Figura 6. Relación de los 5 niveles de vistas CLASP (Owas.org)

Las 24 acciones indicadas en las vistas de implementación y evaluación de actividades para la metodología son descritas en la Tabla 3:

Tabla 3. Relación de actividades sugeridas para el desarrollo CLASP

Actividad	Propósito	Rol	Frecuencia
Direccionar los problemas de seguridad reportados	Certificar que los riesgos identificados sean considerados y direccionados de manera correcta.	Diseñador	Cuando se requiera
Diseños de clase con propiedad de seguridad	Elaborar políticas de seguridad para los campos de datos individuales	Diseñador	Una vez por iteración
Aplicar los diseños de seguridad del proyecto	Reforzar el diseño de aplicaciones mediante la implementación de diseños de seguridad. Identificar los riesgos de seguridad que puedan ser proporcionados por terceros.	Diseñador	Al menos una vez por iteración
Construir guía operacional de seguridad	Proporcionar a los interesados documentación en medidas operacionales de seguridad. Suministrar los documentos necesarios que formalicen el proceso de seguridad de la aplicación.	Implementador	Una vez por iteración
Diseño de los casos de uso indebido	Informar a los interesados sobre los riesgos y las razones de toma de decisiones relevantes que	Especificador de requerimientos	Según sea necesario, sin importar las veces en cada ejecución.

	han sido tomadas para la seguridad.		
Documentación de requerimientos de seguridad	Documentar a nivel de negocio los requisitos funcionales de seguridad.	Especificador de requerimientos.	Una vez por iteración y cuando sea necesario.
Identificar superficie de ataque	Especificar estructuradamente cada punto de entrada (input) del programa. Facilita el proceso de análisis	Diseñador	Recomendado al terminar la fase de diseño, y, con continuidad mediante la elaboración.
Identificar política general de seguridad	Proporciona los requerimientos de seguridad del negocio. Compara el nivel de seguridad de varios productos de la org.	Especificador de requerimientos	Una vez por iteración.
Identificar los recursos y límites de la confianza	Identifica la base estructurada para interpretar los requisitos de seguridad del sistema	Arquitecto	Una vez por iteración
Identificar Roles de usuarios y capacidades de los recursos	Precisa los roles del sistema, así mismo, capacidades y recursos a los que tendrá acceso cada uno.	Arquitecto	Una vez por iteración
Identificar, Implementar y realizar pruebas de seguridad	Encuentra riesgos que no han sido evidenciados en la fase de implementación o, que han sido identificadas en el proceso operativo. Se	Tester	Generalmente, repetidas veces durante cada iteración.

	comporta como mecanismo de captura de fallos.		
Implementar y elaborar política de recursos y tecnologías de seguridad	Aplica las funciones de seguridad en cada especificación.	Implementador	Cuando sea necesario
Implementar interfaz de contratos.	Encuentra los errores a nivel de fiabilidad desde el inicio del proceso	Implementador	Cuando son modificados los métodos o funciones.
Programa institucional de sensibilización de seguridad.	Asegurar que los miembros del proyecto podrán enfrentar los inconvenientes de seguridad con la suficiente eficacia para volverla un objetivo mediante la entrega de cuentas.	Gerente del Proyecto	En marcha
Integrar la seguridad en el origen del proceso de admiración	Automatizar el análisis de seguridad a nivel de los procesos de aplicación y la recolección de datos.	Implementador	Cuando sea necesario
Administrar la revelación de los problemas de seguridad	Comunicar los inconvenientes de seguridad presentados después de puesta en marcha del software.	Gerente del Proyecto	Cuando sea necesario
Controlar las estadísticas de seguridad	Llevar control métrico de los casos de seguridad presentados.	Gerente del proyecto	En marcha

Generar firma del código	Proporcionar a los interesados, la forma de validar el origen y la integridad del software entregado	Implementador	Una vez liberada la aplicación o desarrollo
Hacer análisis de seguridad de los requisitos y diseño (modelado de amenazas)	Identificar las amenazas, mitigar los riesgos y fortalecer los requisitos de seguridad establecidos.	Auditor de Seguridad	Cuando sea necesario
Ejecutar monitoreo de seguridad en la fuente	Encontrar las vulnerabilidades a nivel de software	Auditor de Seguridad	Incremental, al final de cada iteración
Investigar y evaluar la seguridad de las soluciones tecnológicas	Evaluar los niveles de seguridad que se puedan presentar por terceros	Diseñador	Cuando sea necesario
Especificar la configuración y nivel de seguridad de las BD.	Especifica los niveles de seguridad para los usuario y terceros que puedan acceder a la Base de Datos	Diseñador (BD)	Cuando sea necesario
Especificar el entorno operacional	Documentar los supuestos y requisitos de seguridad de manera que pueda ser validado el impacto operacional	Especificador de requerimientos	Cuando sea necesario
Verificar los atributos de seguridad de los recursos.	Confirmar que los procesos de seguridad establecidos en la política, se encuentren en el software	Tester	Una vez por iteración

2.2.5.2 Recursos CLASP

Como apoyo a los procesos de planificación y ejecución de actividades, CLASP presenta una serie de recursos o herramientas que permiten el acceso a los diferentes artefactos.

2.2.5.3 Caso de uso de Vulnerabilidad

Aquí se describe los puntos en los cuales se presenta vulnerabilidad en las aplicaciones de software, con estos casos de uso, se pretende dar al usuario una vista fácil respecto, de esta manera se tendrá clara relaciones causa y efecto.

2.2.6 Touchpoints

Conformada por 7 puntos de control establecidos en el 2004 por la IEEE conocido como el modelo de GaryMCgraw y Cigital, se consideran uno de los tres pilares de la seguridad de software siendo tomada como la unión entre la parte técnica y práctica del desarrollo haciendo énfasis de esa manera en el empleo de buenas prácticas (Mc Graw & Jhon, 2006).

Está compuesta por dos tipos de actividades, unas destructivas (sombrero negro) y unas constructivas (sombrero blanco), en su orden de efectividad se pueden enumerar de la siguiente manera:

2.2.6.1 Fase Revisión de código

Dirigida a la revisión del código fuente con el fin de identificar las posibles vulnerabilidades que puedan afectar el desarrollo del producto. Sin embargo, es entendible que blindar un software a prueba de intromisiones es prácticamente imposible por lo que, se presentan las demás fases de ejecución.

2.2.6.2 Análisis de riesgos de arquitectura

Con el fin de identificar los riesgos posibles en el desarrollo, esta fase enfoca su actividad en el trabajo en conjunto que se presente entre los Arquitectos, diseñadores y analistas, de esta manera podrán ser documentados desde todos los puntos de vista.

La implementación de un proceso de análisis de riesgos y seguridad pueden proporcionar al arquitecto de pautas para el inicio de la construcción, sin apartar el hecho de la posible aparición de amenazas en cualquier etapa del proyecto, estos análisis deben ser constantes en el proceso.

2.2.6.3 Pruebas de penetración

Se realizan intrusiones relacionadas a hacking ético y controlado, con el fin de identificar que posibles inconvenientes se pudieran presentar a futuro.

Su finalidad es implementar procesos para evitar y prevenir los posibles ataques.

2.2.6.4 Pruebas de seguridad basada en riesgos

Es importante tener claro que los ataques maliciosos no son siempre evidentes, para ello las pruebas de seguridad deben contemplar estos posibles puntos basados en la identificación, análisis y seguimiento de riesgo.

2.2.6.5 Casos de abuso

Según Gary Mac Graw, este punto es la forma más directa y cercana para entrar en la mente de un atacante, busca identificar claramente la naturaleza y raíz de los ataques y, la conducta de los sistemas bajo el análisis de ataques definidos. Tiene como finalidad, la identificación de que se debe revisar, proteger y fortalecer, de qué o quién y por cuanto tiempo.

2.2.6.6 Requerimientos de seguridad

Aunque en el inicio de todo proyecto de software se determinan los requerimientos funcionales y no funcionales a percepción de usuario, deben ser construidos procesos enfocados al proceso de seguridad. Estos requisitos deben ser validados y actualizados periódicamente con el fin de reestablecer y renovar los protocolos y procesos establecidos para este fin.

2.2.6.7 Operaciones de seguridad

Igualmente, de manera recurrente y continua, debe ser monitoreado el comportamiento de seguridad del software, este proceso se convierte en la principal forma de defensa de un sistema de información en la actualidad. Su punto de partida se encuentra se encuentra en la identificación de riesgos.

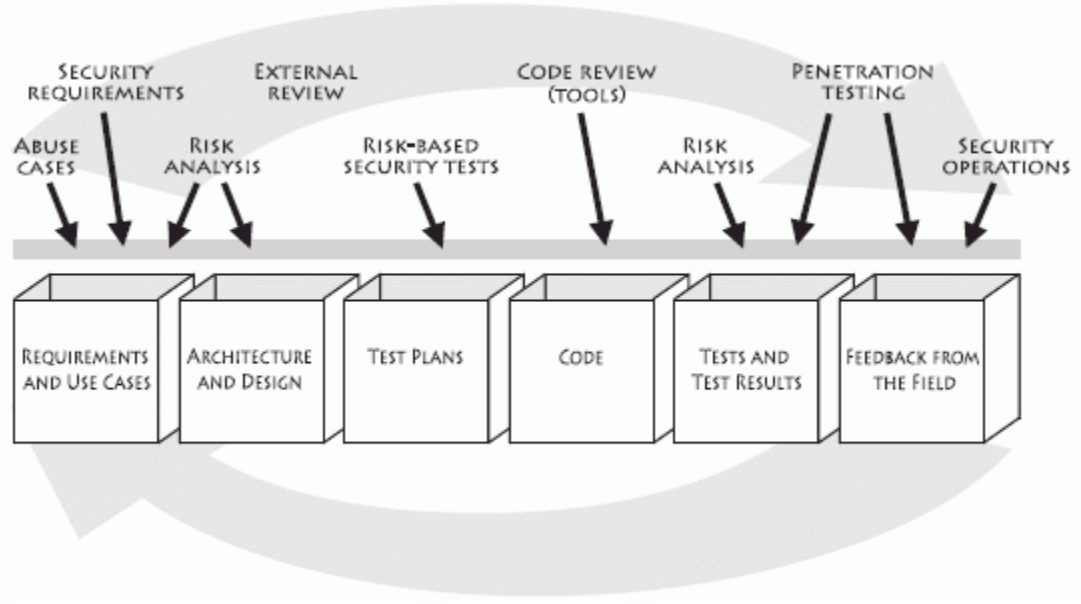


Figura 7- Siete puntos de la Seguridad de Software
(http://www.swsec.com/resources/touchpoints/)

2.2.7 SAMM (Software Assurance Maturity Model)

SAMM facilita (OpenSAMM):

- Evaluación y revisión de las técnicas usadas en los procesos de seguridad.
- Desarrollar programas y estrategias de seguridad con enfocadas en las iteraciones que se puedan definir.
- Definir las actividades que se realizarán en cada procedimiento con el fin de reforzar los procesos de seguridad.

Presenta 4 funciones principales, cada una de ellas compuestas por tres prácticas de seguridad de la siguiente manera:

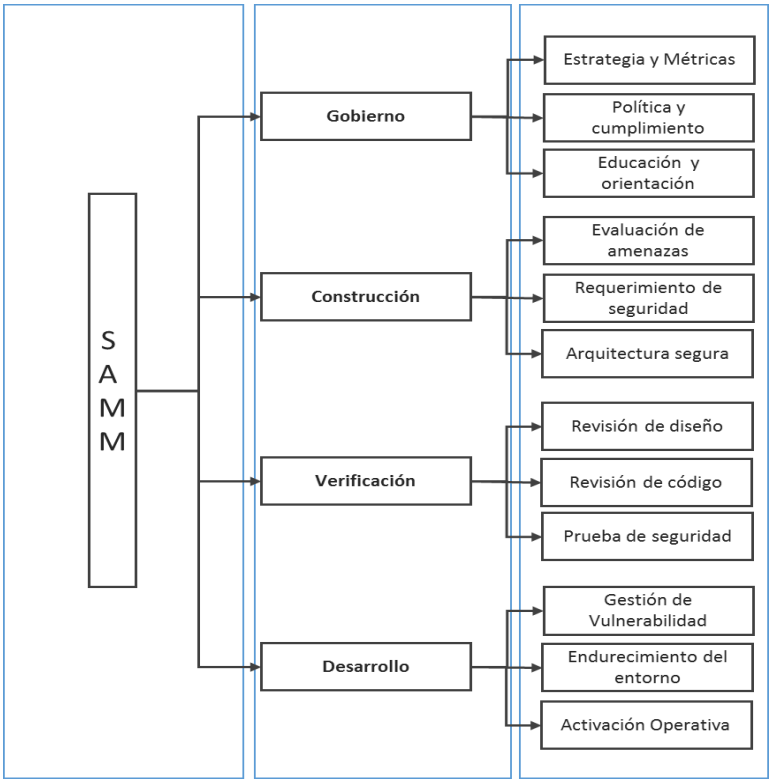


Figura 8. Organización SAMM

Se encuentra una subdivisión que especifica:

1. SAMM (Desarrollo de software): Comprensión inicial y disposición para adoptar la practica
2. Funciones de negocio: Incrementa la eficiencia y eficacia de cada una de las prácticas.
3. Práctica de seguridad: Dominio completo de la práctica. Dentro de esta área, la metodología debe puntualizar los siguientes items:
    a. Objetivo
    b. Actividades
    c. Resultados
    d. Umbrales de satisfacción
    e. Coste
    f. Persona
    g. Niveles relacionados

2.2.8 MAGERIT – Metodología de análisis y Gestión de Riesgos de los Sistemas de Información.

Esta metodología tiene como finalidad, analizar y realizar procesos de gestión de riesgos para los Sistemas de información, creando procesos de iterativos de análisis y tratamiento de los riesgos partiendo de la conservación de los Programas (Gobierto de España.Ministerio de Hacienda y Administraciòn Publica, 2016) . Apoyados en la norma ISO27001, se identifican 4 etapas críticas en el proceso denominadas como proceso PDCA cuyo ciclo se evidencia en la Figura 9:

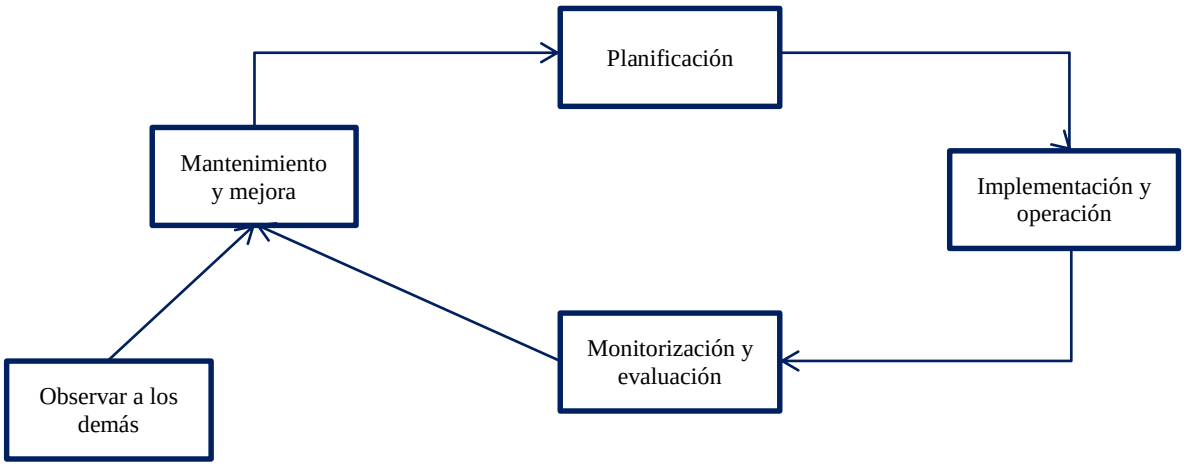


Figura 9. Ciclo PDCA (Plan, Do, Check, Act)

Así mismo, se torna primordial dentro de esta metodología, el proceso de gestión de riesgos, dentro del cual se identifican los siguientes pasos:

- Delimitación del contexto: Facilita la construcción la política de seguridad que se utilizará en la gestión de riesgos. Dentro de sus subprocesos se encuentra: análisis, relación con otras organizaciones en intercambio de información o servicios, y proveedores.
- Identificación de los riesgos: Validará y encontrará los focos de peligro. Esta información es analizada en la siguiente etapa del proceso.
- Análisis de riesgos: Califica y asigna prioridad a los riesgos encontrados.
- Evaluación de los riesgos: Traduce las consecuencias técnicas identificadas a términos del negocio con el fin de tomar decisiones respecto de que riesgos se asumen y se aceptan y cuales se pueden convertir en un problema.
- Tratamiento de los riesgos: Agrupa y determina las actividades específicas que serán trabajadas para dar trámite y cierre a los riesgos que se han identificado.

- Comunicación y consulta: Busca equilibrar el sistema en términos de seguridad y la interoperabilidad organizativa. Para ello, los usuarios y los canales de comunicación son los puntos claves a tener en cuenta.
- Seguimiento y revisión: Propone proceso de control de manera interactiva para validar los avances en disminución de los riesgos encontrados e identificar los que puedan surgir a raíz de este.

2.2.8.1 Método de análisis de riesgo

Entendiendo el proceso de riesgos y su identificación, es primordial la validación de amenazas típicas en un sistema de información. En este contexto se definen los siguientes tipos:

- De origen Natural: Enfocado en accidentes naturales como terremotos, sismos, inundaciones, entre otros.
- Del entorno o industrial: Clasificadas como fallas de electricidad, contaminación, etc
- Defectos de la aplicación: Amenazas presentadas por defectos de diseño o implementación, generando procesos desastrosos en el desarrollo.
- Por intervención Humana de forma accidental: Generalmente presentadas por errores sin intención o por omisión.
- Por intervención humana de forma deliberada: Ataques al sistema con fines de lucro o con el fin de beneficiarse indebidamente de la información o procesos manejados.

Basados en los análisis realizados respecto a Riesgos y Vulnerabilidades que pueden permitir el ingreso de amenazas, se establecen actividades con el fin de minimizar la aparición de amenazas y riesgos.

2.2.9 OWASP: Open Web Application Security Project

Open Web Application Security Project o sus siglas OWASP (Curello Fabio, 2013), es el Proyecto de seguridad en aplicaciones WEB abierto, suministra abiertamente procedimientos de seguridad aplicado a este tipo de aplicaciones que, constantemente presentan nuevas formas de ataques; de esta manera OWASP ha producido una serie de guías para facilitar la adquisición de una base de datos de conocimiento respecto a la seguridad de las aplicaciones:

- Referencia de Escritorio en Seguridad de Aplicaciones de OWASP: Contiene las descripciones y conceptos básicos de la seguridad como: agentes de amenazas, vulnerabilidades, ataques, impactos. Se trata de una guía básica y practica para las demás guías disponibles en OWASP.
- Guías de desarrollo de OWASP: abarca todo el control de seguridad primordiales para el desarrollador de un producto de software, proporcionando un conjunto de controles de seguridad que fortalecerá el Sistema de desarrollo.
- Guía de pruebas de OWASP: Suministra el paso a paso y guía de variación y pruebas validando los procedimientos respecto a la seguridad, con ello se garantiza una revisión exhaustiva del producto.
- Guía de revisión de Código OWASP

Tiene como uno de sus principales pilares, la incorporación de principios relacionados a la gente, los procesos de incorporación de seguridad en un ciclo de vida de desarrollo de software y la tecnología, enfocada esta última en el componente tecnológico.

Su análisis respecto a las vulnerabilidades que afectan las aplicaciones, se divide en tres puntos descritos en la Figura 10:

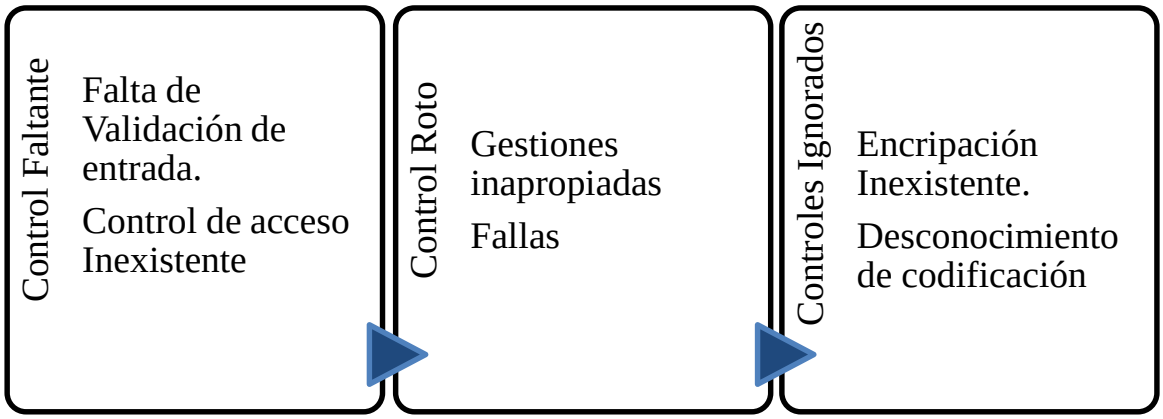


Figura 10. Teoría de Vulnerabilidades

Así mismo, OWASP plantea el Top 10 de los riesgos de seguridad a los cuales se deben enfrentar las aplicaciones WEB, enfocadas al proceso de protección de la información. Las fases descritas en este listado se encuentran descritos en la Tabla 4:

Tabla 4. Top 10 Riesgos de Seguridad aplicaciones WEB (2013)

A1. Inyección: Se presenta cuando datos no confiables son enviados al interprete, ocasionado la intromisión al ejecutar comando no autorizados y permitiendo el ingreso a información sensible. Ejemplo**:** Modificación de una sentencia SQL, por ej: *String query = "SELECT * FROM accounts WHERE custID='" + request. getParameter("id") + "';* Al modificar el *id* en su navegador por *'or'1'='1',* se cambia la consulta y mostrará todos los datos de la tabla. ¿Cómo prevenirlo?: Mantener los datos no confiables alejados de las consultas y comandos. Usar un API seguro que evite el uso de intérpretes, si no se cuenta con ella, se recomienda codificar los caracteres especiales.
A2. Pérdida de Autenticación y Gestión de Sesiones: Las funcionalidades de autenticación se usan generalmente de forma errónea, facilitando el robo de contraseñas, token de sesión, contraseñas y facilitando la usurpación de los usuarios. Ejemplo**:** Falta de definición en los tiempos de cierre de sesión en aplicaciones. Páginas con reescritura de URL donde se guarda información de los últimos ingresos. ¿Cómo prevenirlo?: Cumplir con todos los requisitos de autenticación y gestión de sesiones abiertas.
A3. Secuencia de comandos en sitios cruzados: Se presenta por fallas relacionadas al XSS en el momento en que son tomados datos no confiables y son enviados al navegador WEB. Esto permite que los atacantes puedan ingresar y usurpar la sesión abierta de un usuario o dirigirlos hacia otro sitio en la WEB.

Ejemplo: Los atacantes pueden cambiar en el navegador el parámetro "CC" para direccionar a su página los datos de identificación del usuario.

¿Cómo prevenirlo? Codificar la información y los datos no confiables basados HTML (cuerpo, atributo, JavaScript, CSS, o URL).

Revalidación de entradas positiva, teniendo en cuenta que muchas aplicaciones necesitan aceptación de caracteres especiales como parte de las entradas válidas.

A4. Referencia Directa Insegura a Objetos: Ocurre cuando se expone una referencia de un objeto interno de implementación, en este caso directorios, ficheros, etc. Se requieren chequeos de dichas referencias para evitar la intromisión de atacantes.

Ejemplo**:** Modificación de una sentencia SQL, por ej: *String query = "SELECT * FROM accts WHERE account =?";* PreparedStatement pstmt = connecCon.prepareStatement(query, …); pstmt. setString(1, request.getparameter("acct"));

ResultSet results = pstmt.executeQuery( ); Al cambiar el valor "acct" en el navegador, se pordrá acceder a cualquier cuenta de usuarios.

¿Cómo prevenirlo?: Usar referencias por usuario a sesión. No usar clave del recurso de BD, se sugiere usar una lista de 6 recursos, utilizando una lista del 1-6 para indicar cuál es el valor designado por el usuario.

A5. Configuración de Seguridad Incorrecta: Basada en las todas las configuraciones relacionadas con aplicaciones, Marcos de trabajo, servidores web y de aplicaciones, plataformas, Bases de Datos, etc. Referentes a procesos de seguridad claramente definidos e implementados.

Ejemplo**:** Mantener dentro del servidor de aplicaciones, aplicaciones de ejemplo o que se contienen por default, estas, pueden contener fallos de seguridad y la puerta de acceso fácil para el atacante pues son conocidas.

Instalación automática de la consola de administración del Servidor, cuyas cuentas creadas por defecto no han sufrido modificaciones, esto permite la intromisión sin autorización de las contraseñas configuradas por default para estas cuentas.

¿Cómo prevenirlo?: Ambientes de pruebas (QA) y producción con configuración idéntica, aunque deben contar con contraseñas diferentes.

Uso de arquitectura de aplicación.

Auditoria recurrente respecto a parches o configuraciones para detectar inconsistencias.

A6. Exposición de datos sensibles: Los datos sensibles de los usuarios requieren de mecanismos de encriptación y cifrado para evitar el robo de información clave como tarjetas de crédito, identificaciones, datos de logueo, etc. Las aplicaciones WEB deben contar con este tipo de protección para minimizar los riesgos de usurpación, robos y fraudes.

Ejemplo**:** Cuando el cifrado de este tipo de información delicada es cifrada automáticamente por la base de datos, el proceso de recuperación se ejecuta de la misma manera, permitiendo extraer los datos en formato texto.

¿Cómo prevenirlo?: Se recomienda el uso de claves públicas para cifrar la información y acceder al descifrado con clave privado solamente desde la aplicación backend.

No almacenar datos sensibles innecesariamente

Usar algoritmos de cifrado fuertes, requerir claves de seguridad con estándares fuertes establecidos.

A7. Ausencia de Control de Acceso a Funciones: Verificar las solicitudes de acceso al servidor al ejecutar las funciones por los usuarios.

Ejemplo**:** Los roles que ejecutaran determinada función, no son validados por el sistema por lo cual, cualquier usuario tendría acceso a todas las funciones o acciones, presentando vulnerabilidad y facilidad de acceso a datos sensibles**.**

¿Cómo prevenirlo?: Generar proceso de auditoría de control de acceso a los usuarios y las funciones, se debe restringir el acceso inmediatamente es identificada la falencia del rol permitido

A8. Falsificación de PeCciones en SiCos Cruzados (CSRF): La victima del ataque es inducido a enviar una petición tipo HTTP falso, al suceder mientras se encuentra logueado, son enviados los datos de sesión a una aplicación Web en la cual, el atacante podrá hacer solicitudes a esta página con los datos de autenticación de la víctima.

Ejemplo: Pueden ser construidas peticiones de envió de dinero de una cuenta a otra, posteriormente el atacante inserta en una etiqueta de imagen (iframe) su ataque que han sido almacenados en diversos lugares almacenados por él. Al ingresar a alguno de estos sitios ya autenticado, autorizará la petición del atacante.

¿Cómo prevenirlo?: Requerir que el cliente vuelva a autenticarse probando de esta manera que se trata de un usuario legítimo.

A9. Utilización de componentes con vulnerabilidades conocidas: Algunos frameworks y librerías funcionan con todos los privilegios, si en alguno de ellos se

presentan vulnerabilidades, puede ser usado para acceder a las aplicaciones ampliando las posibles incidencias.

Ejemplo**:** Framework Apache CXF Autentication Bypass, no cuenta con token de identidad, podría ser agregado cualquier servicio Web con todos los accesos.

Spring Remote Code Excecution, el componente "Expression Lenguaje" facilito la ejecución de código por los atacantes.

¿Cómo prevenirlo?:Crear parces de vulnerabilidad de las versiones más antiguas, no solo corregir el error en las siguientes versiones.

A10. Redirecciones y reenvios no validados: Re direccionamiento del usuario a paginas mediante procesos no confiables, pueden concluir en la asignación de sitios de phishing o malware.

Ejemplo**:** En caso de vulnerabilidad en el proceso de envió a otra página, el atacante podrá crear una URL falsa, en al cual se instalará el phishing.

¿Cómo prevenirlo?: Podría evitarse suprimiendo los procesos de reenvió a otras páginas o, no utilizar parámetros que puedan ser manipulados por el usuario.

2.3 Marco Legal

Para garantizar que el desarrollo de este proyecto cuenta con los soportes legales respecto a las normatividades ligadas al tema de desarrollo, la ley colombiana, cuenta con reglamentos establecidos con el fin de suministrar herramientas, en cuanto a ataques informáticos o calidad intelectual se refiere.

A raíz del auge y crecimiento tecnológico en Colombia, los productos de software y el creciente involucramiento de la sociedad en dicho ámbito, se ha visto el incremento desmedido así mismo de ataques informáticos; basados en esto, el país ha visto en la necesidad de construir una normatividad que sea el instrumento para enfrentar y detener estas intromisiones.

Entre las actuales leyes en Colombia encontramos la *Ley 1273 Enero 2009 (Ley de delitos informáticos en Colombia)* la cual establece nuevos códigos penales referentes a la protección de la información y las penas en las que se puede incurrir relacionado con delitos informáticos (Gandini I., Isaza A, Delgado A. Delta Asesores). En ella, se tiene en cuenta: Acceso abusivo a un sistema informático, Obstaculización ilegitima de sistemas de informáticos o redes de comunicación, Interceptación de datos informáticos, Daño informático, Uso de software malicioso, Violación de datos personales.

Mediante su sanción, se clasificaron algunas acciones ligadas al manejo de datos personales como delitos, encaminada a la protección de la información sensible y que se encuentra cobijada bajo esta ley en el *Código Penal Colombiano en el "Titulo VII – De la Protección de la Información y de los Datos".* Otra ley que tiene como finalidad definir mediante reglamentos, el acceso a los mensajes de datos al igual que su uso particular,

teniendo en cuenta procesos de comercio electrónico y la aplicación de firmas digitales, estableciendo de esta manera las entidades de certificación aprobadas es la *Ley 527 de 1999*, la cual indica los Ámbito de aplicación (Barco, 1989).

Basados en la necesidad de reglamentar el proceso de derechos de autoría del software, mediante el decreto 1360 de 1989, los desarrolladores podrán registrar en el Registro nacional de derechos de autor, todo aquel producto de software el cual, se convierte en la arma para defender los productos lógicos; este decreto se apalanca en la *Ley 23 de 1982 de Derechos de Autor*, y consta de 8 artículos en los cuales es considerado el software como "Creación Propia del Dominio Literario" e identifica el proceso a llevar a cabo para el registro un sistema lógico.

El *Documento Conpes 3854 – Política de Seguridad Digital*, ha sido establecido por el El consejo Nacional de Política Económica y Social (Conpes), el cual, es el organismo encargado de la asesoría nacional respecto a los factores económicos y sociales del país, realizando proceso de análisis y aprobación de documentos encaminados al desarrollo de diferentes ámbitos nacionales. Este documento tiene como finalidad el involucramiento de los altos niveles del gobierno con el fin de establecer un conjunto de principios y acciones fundamentales basado en una visión estratégica que fortalezca la gestión de riesgos a nivel de la seguridad digital en Colombia. El documento fue dado a conocer en el *Segundo Foro de Ciberseguridad para Ambientes Industriales e Infraestructura crítica*, con el que se espera afinar los estándares de seguridad al igual que plantea nuevas actividades tanto para la sociedad como para el Gobierno Nacional. (Consejo Nacional de Política Económica y Social, 2016).

2.3.1 Licenciamiento GPL V3

El 29 de Junio del 2007 fue publicada la 3ra Versión de la licencia GPL (Licencia Pública General de GNU), su uso, garantiza que la nueva versión creada, contará con el mismo tipo de licenciamiento del original (Free Software Fundation, 2016), es así como, al evaluar las diferentes herramientas involucradas en el desarrollo de este Producto de software, encontramos que tanto MySql presentando un doble licenciamiento GPL y Licencia comercial por Oracle Corporation, como Laravel con licencia MIT, que permite reutilizar software dentro del software propietario facilitando la integración con GNU pero no al contario, y Jquery con licenciamiento dual MIT y GNU, cuentan con licenciamiento open source de este tipo o compatible con él, por lo cual, y teniendo en cuenta las implicaciones de esta licencia, el Sistema de Control de Versiones para el Desarrollo de Software Seguro, adoptará este licenciamiento en el que, se permite compartir y modificar de ser necesario el Producto de software, permitiendo al autor del sistema conserva los derechos de autor o copyright aunque, permita el proceso de distribución y cambios. Dado que el proyecto puede ser modificado y adaptable para otros requisitos de software, la licencia GNU, hace que, la toma parcial o total y el producto resultante de esta modificación deberán contar con el mismo tipo de licenciamiento.

Para tener claridad acerca de los detalles del tipo de licenciamiento de software libre usado para el proyecto, validar Anexo I – Guía GPLv3

Capítulo 3

Sistema Propuesto

3.1 Selección de la Metodología

Teniendo en cuenta la cantidad de integrantes del proyecto y que será ejecutado a corto plazo, la metodología ágil utilizada para el desarrollo del Sistema de Control de Versiones para el Desarrollo de Software Seguro fue Xtreme Programming (XP) (Letelier & Penadés, 2006). Al utilizar esta metodología, se encamina su énfasis en la adaptabilidad del desarrollo más que a la previsibilidad, teniendo en cuenta los posibles cambios de requerimientos que se presenten en el camino.

Dentro de las prácticas que recomienda este tipo de metodología, el desarrollo de este proyecto tiene como base:

- Juego de la planificación: Se mantendrá comunicación con el usuario con una alta frecuencia. Serán realizadas las estimaciones técnicas respecto a las "historias de usuario" o requerimientos funcionales y no funcionales. El cliente establecerá las fechas de entrega.
- Entregas pequeñas: Se hará entrega de pequeñas partes del desarrollo, aunque no expresen la funcionalidad completa del Sistema de Información, con esto, se cumplirán con los adelantos planeados con el usuario.
- Metáfora: Sera construida en conjunto la historia de usuarios que defina como deberá funcionar.

- Diseño simple: Sera creado un diseño simple que cumplas con las especificaciones del usuario.
- Pruebas: El usuario establecerá las pruebas unitarias que podrán ser ejecutadas en el transcurso de las entregas programadas.
- Refactorización: Actividad de reestructuración para facilitar y depurar el código empelado en la aplicación.
- Programación en parejas: La producción del código será ejecutado en los integrantes del grupo de trabajo.
- Propiedad colectiva del código: Los programadores involucrados podrán realizar procesos de mejora en el código.
- Integración continua: Cada parte de código será integrado al todo para realizar las validaciones.
- Cliente in-situ: La comunicación oral entre los programadores y el cliente, proporciona un ambiente claro y sencillo a la hora de validar y aclarar dudas referentes al funcionamiento esperado.

Se ha identificado el siguiente flujo de procesos para graficar el comportamiento esperado por el cliente en el Sistema de Control de Versiones para el desarrollo de Software seguro planteado en la Figura 11:

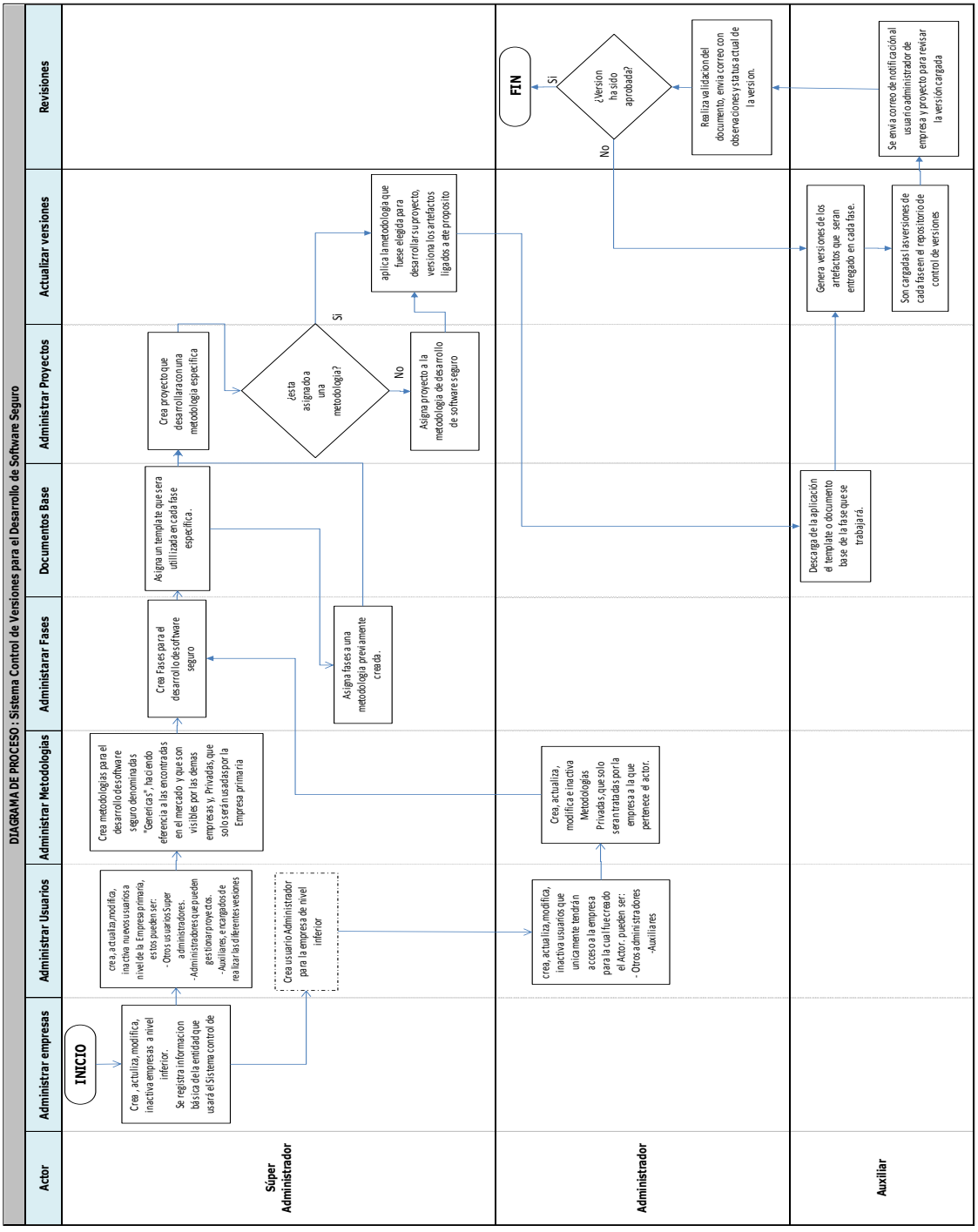


Figura 11. Diagrama de Procesos

La caracterización del recurso que fue implementado en el desarrollo del proyecto se identificará a continuación:

Recurso Humano:

Usuario/cliente: Persona o entidad que presenta un problema o necesidad que debe ser suplida.

- Punto de contacto que define los requerimientos que debe cumplir el software desarrollado.
- Interactúa de manera constante con el equipo de proyecto.
- Ejecuta pruebas de validación planeadas a los prototipos entregados
- Da el aval y aprobación en cada una de las etapas.
- Realiza prueba de aceptación respecto al software entregado.

Programador: Desarrollador de la aplicación, de acuerdo a las necesidades e historias de usuario determinadas en el proceso.

Gestor: Facilita la comunicación sirviendo de puente entre el usuario y el programador.

Tutor: Para el caso del presente proyecto, se incluirá esta figura que hará las funciones de usuario o cliente.

Los roles de Programador y Gestor podrán ser llevados por todos los miembros del equipo en determinado caso, con reuniones periódicas con el usuario, para validar y atacar los posibles cambios o riesgos que se presenten durante el proceso.

Recurso Técnico:

En el proceso serán utilizados equipos tipo PC de cada integrante del equipo en el cual se llevará a cabo el proceso de programación y documentación.

- Será usado paquete de office incluyendo Microsoft Project 2013 para llevar el control del proyecto.
- Framework Laravel 5.2 donde será desarrollado la aplicación tipo web con código PHP.
- MySQL como sistema gestión de bases de datos.

A continuación, en la Tabla 5 se identifican algunos de los artefactos de seguridad que se identifican para dos de las metodologías del desarrollo de software en su ciclo de vida general: Entendiendo que el desarrollo del proyecto es corto, se hace uso de metodologías agiles integrándoles buenas practicas que permitan la implementación de la seguridad a partir de los requerimientos no funcionales. Entre las metodologías más usadas para este tipo de proyectos tenemos:

Tabla 5. Buenas prácticas de seguridad en Metodología de desarrollo de Software

Metodología	Fase	Artefacto o buena práctica
Xp	Planeación	El cliente, el gestor y el jefe de proyectos, plantean los requerimientos no funcionales de seguridad a ser tenidos en cuenta durante el desarrollo de la aplicación. Historia de mal uso construidas en conjunto con el cliente y el equipo de desarrollo.
	Diseño	El equipo de desarrollo diseña las Tarjetas CRC – Dentro de las responsabilidades se incluyen elementos de seguridad a ser tenidos en cuenta en el desarrollo de la aplicación durante la iteración correspondiente.
	Codificación	Entrega de los módulos desarrollados que para el caso de estudio incluirán niveles de seguridad. El encargado del seguimiento, debe verificar la implementación de seguridad en los módulos desarrollados de acuerdo a las Tarjetas CRC.

		El entrenador y el consultor apoyan al equipo de desarrollo ante las dificultades encontradas al momento de implementar la seguridad en la construcción de la aplicación. El equipo y los demás miembros presentan al cliente avances o entregables de la aplicación incluyendo de los requerimientos funcionales y no funcionales, lo cual permite el cierre de la fase o una nueva iteración para incluir elementos pendientes.
	Pruebas	El encargado de las pruebas verifica mediante las correspondientes pruebas, que estas responden a la seguridad planeada en los requerimientos no funcionales. En la reunión de cierre el jefe de proyecto verifica el cumplimiento de los requerimientos Funcionales y no funcionales y la implementación de estos en la aplicación.
	Modelado de negocio	Análisis arquitectónico, sistema actual e interfaces actuales.

Scrum		Product Ower es el responsable inicialmente sobre la inclusión de los requerimientos no funcionales como la seguridad dentro de la aplicación, esta responsabilidad la puede delegar al Scrum Master o a un miembro del equipo de desarrollo con conocimientos en el tema.
	Requisitos	Requerimientos de seguridad del usuario.
	Análisis y Diseño	Diseño del Sistema, evaluando vulnerabilidades. Sugerencias de seguridad por parte del desarrollador
	Implementación	Plan de implementación y validación del sistema. En las reuniones diarias tener en cuenta: Que se ha realizado, que dificultades se han presentado y que se piensa hace.

Pruebas/Despliegue	En reunión de entregas del sprint se manifiesta sobre los niveles de seguridad incluidos en los módulos desarrollados. Para los casos en que se no se logró integrar niveles de seguridad en el sprint correspondiente, este formará parte de un nuevo sprint. Reunión de cierre, el equipo expresa las buenas prácticas implementadas en seguridad durante el desarrollo de la aplicación las lecciones aprendidas, así como, los aspectos a mejora para futuros proyectos. Product Owner debe validar que los requerimientos no funcionales como la seguridad, se implementaran acorde a lo establecido con el cliente para la continuidad del negocio.

3.2 Fases de la Metodología Ágil XP (Xtreme Programming)

Basados en la metodología ágil trabajada (Xtreme Programming), han sido identificadas las fases que la componen y así mismo, se identifica el detalle de cada una aplicada al Sistema de Control de Versiones para el Desarrollo de Software Seguro:

3.2.1 Planeación

3.2.1.1 Historias de Usuario

Entendiendo la funcionalidad del software desarrollado, fueron recopilados los requerimientos funcionales y no funcionales del usuario en las denominadas *Historias de Usuario.*

En la extracción de esta información, ha sido utilizado el formato establecido por la metodología implementada para el desarrollo, a continuación, se encuentra el ejemplo de la Historia de Usuario referente al ingreso de los usuarios en la Tabla 6:

Tabla 6. Formato Historias de usuario

Historia de usuario	
Número: 1	**Nombre**: Login - Ingreso al sistema
Usuario: súper Administrador, Administrador y Auxiliar	
Modificación de historia Nº: 0	**Prioridad del negocio:** Alta
Descripción: Solo podrán ingresar al sistema los usuarios previamente creados por los usuarios Súper Administradores y Administradores, en caso de no estar creado, se generará mensaje de error e impedirá el proceso.	
Observaciones: Por política de seguridad, no se permite el ingreso de usuario no registrados.	

Definiciones:

- Número: Enumera y asigna un numero de referencia a cada historia de usuario.
- Nombre: Determina el nombre general referente a la historia de usuario creada.
- Usuario: Identifica el (los) usuarios o actores que ejecutarán el proceso indicado.
- Modificación de Historia N°: Indica la cantidad de modificaciones que se han desarrollado en la historia al terminar las iteraciones planeadas.
- Prioridad del Negocio: Cataloga la importancia indicada por el usuario para la historia, pueden ser:
    - Alta: Asignada a procesos infaltables en el proceso que pueden representar vulnerabilidades en el sistema y que se pretende disminuir.
    - Media: Procesos importantes para el usuario pero que, no representa riesgo de generación de vulnerabilidades o, que representa procesos No críticos para el cliente.
    - Baja: Determinado como requerimientos funcionales agregados al proceso crítico pero que no representan riesgo, ejemplo de visualización de interfaces, entre otros.
- Descripción: Indica el detalle del requerimiento funcional o no funcional solicitado por el cliente para construir la historia y que, se debe cumplir en el desarrollo del proyecto.
- Observaciones: Aclaraciones o factores externos que debieran ser tenidos en cuenta.

Como resultado, tenemos el detalle encontrado en el Anexo B – Historia de Usuario y el recuento en la Tabla 7:

Tabla 7. Registro Historias de usuario

Tipo de Historia de Usuario	Cantidad Total Registradas	Cantidad Total Modificadas
Historias Funcionales	17	8
Historias No Funcionales	15	0

3.2.1.2 Plan de entregas

La estructuración del plan de entregas para el desarrollo del proyecto, ha sido establecida mediante el formato encontrado en la Tabla 8:

Tabla 8. Relación de iteraciones e Historia de Usuario

Iteración	Fecha	Orden de la Historia de Usuario	Duración de las Iteraciones
1ra	16 Marzo 2016	✓ Administración de Usuario ✓ Administrar Menú Roles ✓ Login - Ingreso al sistema	2 Semanas

Definiciones:

- Iteraciones: Cada una de las reuniones de plan de entregas definidas para tener interacción con el cliente y entrega de actividades propuestas.
- Fecha: Fecha estipulada para entrega de avance.
- Orden de la Historia de Usuario: Identifica en qué orden serán desarrolladas las historias de usuario definidas para cada iteración.

- Duración de las iteraciones: Alcance en semanas que representa el desarrollo y el fin de cada iteración.

Como punto adicional, se identifica el detalle de las tareas que se deben desarrollar en el tiempo establecido y para la duración planeada. A continuación, se establece un ejemplo del desarrollo ejecutado en la Tabla 9:

Tabla 9. Detalle de actividades por iteración

Iteración	Orden de la Historia de Usuario	Tarea
1ra	✓ Administración de Usuario ✓ Administrar Menú Roles ✓ Login - Ingreso al sistema	1. Diseño Interfaz de Usuario 2. Diseño de menú de usuario respecto a roles 3. Creación de usuario Súper administrador 4. Asignación de roles 5. Validación de ingreso a usuario. 6. Validar función Listar Usuarios. 7. Crear y validar función Actualizar Usuarios. 8. Crear y validar función Consultar Usuarios. 9. Crear y validar función Actualizar Usuarios. 10. Crear y validar función Modificar Usuarios. 11. Crear y validar función Inactivar Usuarios. 12. Pruebas.

Entendiendo las necesidades del usuario y las Historias de Usuario recopiladas en la primera actividad, ha sido definido el detalle de esta planeación en el anexo C – Plan de entregas.

3.2.1.3 Velocidad de Proyecto:

Teniendo en cuenta la recopilación las historias de usuarios, las expectativas y la complejidad que se pueda presentar en el desarrollo de cada una de las actividades planeadas en el plan de actividades, el proyecto es desarrollado en 28 semanas (7 meses) abarcando las primeras entrevistas y reuniones con el cliente.

3.2.1.4 Iteraciones:

Durante la ejecución de cada iteración, se presentaron validaciones para determinar si la entrega realizada cumplía con las especificaciones del cliente, por lo cual, se presentaron correcciones y revisiones posteriores para garantizar el correcto funcionamiento del Sistema. De esta manera, la funcionalidad del Sistema Control de Versiones se encuentra dividida en módulos que se expresan en la Tabla 10:

Tabla 10. Definición de módulos y entregas en cada iteración

Módulo	**1ª Iteración**	**2a Iteración**	**3ª Iteración**	**4ª Iteración**	**5ª Iteración**	**6ª Iteración**
Usuario	V1		V2			Terminado
Empresa		V1				Terminado
Metodología				V1	V2	Terminado
Fases				V1	V2	Terminado
Documentos				V1		Terminado
Proyectos				V1		Terminado
Versiones					V1	Terminado

Definición:

- V1: Identifica la primera entrega del módulo terminado acorde a la planeación.
- V2: Segunda entrega de módulo con las modificaciones realizadas después de la entrega generada al cliente.

3.2.2 Diseño

3.2.2.1 Diseño Simple

La metodología propone la construcción de diseños simples y agiles que de igual manera garanticen los procesos de seguridad mínimas para el desarrollo, generando así mismo, un sistema entendible para el cliente, fácil de instalar y manejar.

Al ser un desarrollo a la medida, ha sido definido un enfoque amigable e intuitivo para el usuario, enfocado en procesos de seguridad para su acceso y la información que se tenga en el repositorio. Fue desarrollado mediante una arquitectura Modelo Vista Controlador separando los datos de las interfaces de usuario, para facilitar y brindar las características de escalabilidad y mejoras que sean necesarias.

3.2.2.2 Glosario de términos

Con el fin tener claridad respecto a los diferentes términos usados para el desarrollo del proyecto, han sido identificadas las definiciones claves a tener en cuenta para este fin:

Secure Software Development Life Cycle (S-SDLC): Ciclo de vida para el Desarrollo de Software Seguro que define los procesos seguros a implementar en el desarrollo de Software. (Cigital, Inc, 2016)

Sistema control de versiones (VCS): Herramientas que facilitan el proceso de control de un producto, respecto a una versión, adición o modificación del mismo. Generalmente utilizada para controlar los cambios en codificación, permitiendo llevar una bitácora de los estados anteriores y actuales que ayudan a la corrección de errores.

Metodología: Procedimiento que define el camino a seguir para ejecutar determinado proceso. Determina el paso a paso definido.

Fase: Identifica el estado actual en el que encuentra la metodología, precisa el paso o subproceso que se debe ejecutar para cumplir con el requerimiento.

Artefacto: Documento o entregable que se realiza como resultado de la culminación de una fase aplicada en una metodología, deben pasar por proceso de aprobación y validación para darse por cerrado en el proyecto.

Versión: Cambio realizado sobre un documento, producto o proceso, que se genera con relación a la inicial pero que, sufre grandes o pequeñas modificaciones con el fin de obtener aprobaciones.

Laravel: Framework que permite la construcción y creación de aplicaciones con lenguaje de programación PHP, es de código abierto y facilita la simplicidad y elegancia en el desarrollo. Proporciona encriptación Hash seguro con el cual se realiza el encriptamiento de contraseñas de los usuarios, generando una semilla a partir de la cual, se lleva a cabo el proceso, el método de verificación igualmente permite validar si un string corresponde a un hash determinado. (Laravel, 2016)

Base de datos: Bancos o repositorios de información que presenta interrelación entre tablas de almacenamiento de datos acorde a las especificaciones y necesidades del Sistema de Información que sea planteado.

Repositorio: Localización centralizada en la cual, se almacena la información en este caso digital, que generalmente se caracteriza como una base de datos.

MySQL: Sistema gestor de Bases de Datos (Database Management System, DBMS) para bases de datos relacionales, es una aplicación que facilita el gestionamiento de archivos .sql. (Oracle, 2016)

PHP: Lenguaje de programación flexible de alto rendimiento interpretado por un servidor WEB que genera una WEBSIDE resultando. (PHP Corporation, 2016)

JQuery: Biblioteca JavaScript que simplifica la interacción con documentos HTML, permite la manipulación de eventos y el desarrollo de animaciones, facilitando la inclusión AJAX a las páginas WEB. (jQuery Fundation, 2016)

JQuery UI (User Interface): Biblioteca de componentes que agrega plug-ins, widgets y demás efectos visuales para aplicativos WEB. (jQuery Fundation, 2016)

Bootstrap: Framework de marquetización de HTML, CSS, JS que apalanca el desarrollo de aplicativos WEB en móviles. Facilita el desarrollo front-end web de una manera más ágil y amigable. (Bootstrap, 2016)

Riesgo: validación del grado de vulnerabilidad que se presenta en un Sistema de Información, revisando el impacto y la probabilidad de que este se presente generando daños a la organización o desarrollo.

Análisis de Riesgo: Proceso de estimación donde se definen procesos de tratamiento para los riesgos evidenciados y, de esta manera identificar los planes de mitigación que se pueden ejecutar para acabar con ellos.

Vulnerabilidad: Característica que determina que tan alto es el grado de exposición que se tiene respecto a las amenazas que lo rodean, evidencia la incapacidad de resistencia cuando se presenta o se intenta reponer de un ataque; Está directamente relacionada con la seguridad de la información puesto que, la debilidad puede llevar a perjudicar aspectos primordiales que de cumplir un Sistema de Información: confidencialidad, integridad, disponibilidad y autenticidad de los datos. (Ministerio de Educación, Cultura y Deporte. Gobierno de España, 2012)

Amenazas: Las amenazas son definidas como corrupciones metódicas, a medida que aumentan a nivel técnico e informático, es vital el desarrollo de mecanismos que disminuyan la vulnerabilidad y de esta manera mitigar los riesgos, para ello ha sido desarrollado un proceso de modelado de amenazas. (Markus Erb, s.f.).

Modelado de amenazas: El modelado de amenazas es el proceso que facilita la compresión de las diferentes agentes externas que pueden poner en riesgo un sistema de Información, permitiendo preparar las defensas necesarias que garantizarán las fases de diseños, implementación, revisión y proceso de pruebas. Mediante la implementación de este procedimiento, son aclaradas las políticas que facilitarán la revisión del diseño o modelado arquitectónico para identificar y corregir los posibles inconvenientes de seguridades actuales y que, además se puedan presentar.

Esta metodología es planteada por Microsoft donde expone 5 pasos fundamentales:

- Identificar los objetivos de seguridad: Con la claridad y entendimiento de cada objetivo, será posible medir el esfuerzo que se impartirá en el proceso.
- Crear descripción general de la aplicación: Teniendo como punto de partida la visión general y clara en referencia a la aplicación que se implementarán, será posible identificaran las amenazas más comunes que se presentan.
- Descomponer la aplicación: Se definirán las funcionalidades, módulos y arquitecturas susceptibles a irrupciones de impacto considerable.
- Identificar amenazas: Basados en el detalle del Sistema de información, podrán ser comprobadas las amenazas mayores acorde al contexto.
- Identificar vulnerabilidades: Revisar cuidadosamente cada una de las capas de aplicación para que puedan ser definidos los focos de ataque y los posibles puntos débiles.

Al tener claridad sobre las posibles amenazas, es importante definir el nivel de exposición que tendrá o tiene un sistema de información, para ello, se define como el riesgo

a la relación que existe entre la probabilidad de que ocurra el ataque y el posible daño potencial o impacto general que se pueda tener. (Microsoft, 2016)

Riesgo = Probabilidad * Daño Potencial (Impacto)

SSI (Software Security Institute): Encaminado a suministrar información, entrenamiento y liberaría de investigación para protección de aplicaciones incluyendo las WEB. (SANS Institute, 2008)

GSSP (Secure Software Programmer): Encargado de certificar a los usuarios respecto a los requisitos de codificación que se deben implementar de forma segura. Cuenta adicionalmente con una aplicación online que permite realizar la validación. (SANS Institute, 2008)

SSL secure Sockets Layer: Capa de puertos seguros basados en protocolos criptográficos encargados de proporcionar comunicaciones seguras por la red, comúnmente para interacciones con Internet al igual que su sucesor es el TLS (Transport Layer Security) o, Seguridad de la capa de transporte.

3.2.2.3 Riesgos

Como proceso fundamental en el desarrollo del proyecto de Software, deben ser establecidos los parámetros de identificación, análisis y mitigación de riesgos. Se identificarán responsables del seguimiento y planes de acción para mitigar su impacto, garantizando el seguimiento a los factores que puedan incurrir en el incumplimiento de los objetivos establecidos para el proyecto.

***Identificación*:** Este proceso incluye el detalle de todos los requerimientos, incluyendo propietario y fecha de cierre, Figura 12.

- Identificados desde la planeación del proyecto.

Figura 12. Tipos de Riesgos

Clasificación: Amplia el valor de la comprensión, la documentación y la información sobre los riesgos a nivel de proyecto, tratando de asignar a cada riesgo una escala numérica, ayudando a generar:

- Impacto real
- Priorización de las actividades
- Respuestas de reducción de riesgos identificados.

Al realizar esta clasificación, es importante tener en cuenta la probabilidad (posibilidad de que ocurra) y el impacto (afectación de cada riesgo). De esta manera es identificada la exposición para el proyecto. En el caso de este proyecto de software, se ha definido en la Tabla 11, la relación de probabilidad e impacto de un riesgo:

Tabla 11. Relación Probabilidad e Impacto del Riego.

Probabilidad	***Impacto***
81% - 100% Altamente probable que ocurra.	**9 - 10** Evento catastrófico que implica que el proyecto se realice o se suspenda si ya se ha iniciado. En ningún caso se obtendrán los objetivos propuestos.
61% - 80% Es mayor la probabilidad de que ocurra a que no ocurra	**7 - 8** Puede causar la interrupción del proyecto. En todo caso será necesario reprogramarlo y recalcular los recursos.
41% - 60% Existe igual probabilidad de que ocurra a que no ocurra.	**5 - 6 Puede** causar retrasos o trabajos adicionales. Generalmente se mitiga asignando recursos adicionales
21% - 40% Poco probable que ocurra.	**3 - 4 Es** probable que cause algún atraso o trabajo adicional. En todo caso se podría mitigar con el plan de contingencia
1% - 20% Muy improbable o casi imposible que ocurra	**1 – 2** Causaría dificultades que no comprometen los resultados ni presupuesto del proyecto.

Con la anterior información, será determinado el grado de exposición (Figura 13), de la siguiente manera:

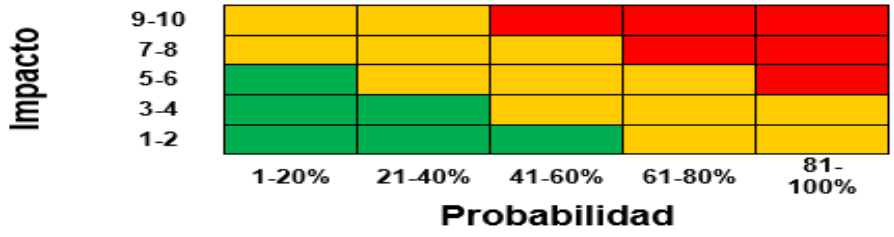


Figura 13. Asignación de exposición.

Los riesgos identificados hasta el momento en el desarrollo del proyecto se encuentran relacionado en la Tabla 12, sin embargo, como ya se ha indicado, el análisis de riesgo debe ser un proceso iterativo ya que pueden presentarse en cualquier etapa e incluso en la implementación.

Tabla 12. Identificación de riesgos

Riesgo	***Impacto***	***Probabilidad***	***Riesgo***
Producto final con diferencias respecto a las Historias de usuarios	9	10%	Medio
No disponibilidad de Internet	9	60%	Alto
Inconveniente en la carga de Versiones	10	20%	Medio
Visualización de información no relacionada al perfil y la empresa del usuario.	10	5%	Medio

Plan de contingencia:

Ha sido definido las siguientes contingencias teniendo en cuenta los riesgos identificados, se debe realizar el proceso de identificación y análisis acorde a lo establecido.

1. Producto final con diferencias respecto a las Historias de usuarios: Iniciar el proceso de validación con el Cliente de las historias de Usuario recopiladas en el proceso y las pruebas de aceptación realizadas.
2. No disponibilidad de Internet: El usuario puede realizar la conexión a su equipo desde modem telefónico, cableado o Wifi.
3. Inconveniente en la carga de Versiones:
    - Descargar los documentos Base, realizar el diligenciamiento de los artefactos.
    - Realizar validación y corrección de código.
4. Visualización de información no relacionada al perfil y la empresa del usuario:
    - Realizar validación de perfiles del usuario.

3.2.2.4 Re factorizar

El proceso de refactorización, permite la revisión del código implementado con el fin de buscar posibles métodos de optimización, esto en el caso de rehusar código ya creado, con el fin de evitar inconvenientes futuros. Para el desarrollo de este proyecto, no fue reutilizado código.

3.2.2.5 Tarjetas CRC (Class, Responsibilities and Collaboration)

La estructuración en la fase de diseño para la metodología XP, se presenta mediante la vinculación de los diferentes módulos y la especificación de sus funciones, su construcción hace alusión al diagrama de Clases usado en una metodología de desarrollo de software común. La aplicación de estas tarjetas suministra un enfoque centralizado respecto a las clases, sus responsabilidades y las demás clases que pueden colaborar con la descrita. Estas relaciones se identifican mediante el formato definido en la Tabla 13:

Tabla 13. Formato diligenciamiento tarjetas CRC

MODULO USUARIOS	
Funcionalidad	**Colaboraciones**
Listar Usuarios	Empresas
Actualizar Usuarios	
Consultar Usuario	
Modificar Usuario	
Inactivar Usuario	

Definiciones:

- Funcionalidad: Define cada una de las funciones que puede cumplir el modulo identificado.
- Colaboraciones: Determina los módulos con los cuales tiene relación.

Para tener mayor detalle respecto a las tarjetas CRC, validar el Anexo D- Tarjetas CRC en el documento.

Adicional a este entregable definido por la Metodología XP, se encuentra el Anexo E – Diagrama Relacional

3.2.3 Codificación

La metodología XP sugiere un esquema de desarrollo conjunto en el que los programadores puedan interactuar en todas las tareas planeadas, de esta manera, cualquiera podrá modificar y revisar el código de otro integrante del equipo.

Los riesgos referentes a la intervención de diferentes programadores en un mismo código, se minimizan debido a que se plantea el proceso de pruebas de calidad y funcionalidad en cada paquete de entrega, de esta manera, se garantiza que se cumpla con las historias de usuario y los tiempos de entrega establecidos.

En el transcurso del proyecto, fue definido proceso de validación para identificar las posibles fallas de funcionalidad, de codificación, consulta, query etc.

Acorde a las diferentes iteraciones y requerimientos propuestos, el sistema de información es creado tipo WEB con aplicación de un framework laravel 5.2 y lenguaje de programación PHP e inclusión de bibliotecas como JQuery de Java Script y JQuery UI incluyendo la interacción de técnicas de desarrollo WEB como AJAX (Asynchronous JavaScript And XML), con esto, se estableció comunicación Asíncrona con el servidor en segundo plano mientras, los procesos de ejecución son realizados en el navegador del usuario; de esta forma, es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones. Mediante el uso del laravel 5.2, fue construido un esquema de menús y submenús que permiten navegar y ejecutar procesos en la aplicación, acorde al rol o perfil que el usuario desee. Igualmente, fue implementado Bootstrap como framework de marquetización con el cual, se facilitar la interacción con aplicativos móviles.

3.2.4 Pruebas

Adicional a las pruebas unitarias realizadas por los programadores en la fase de codificación, la fase que define el desarrollo ejecutado en las anteriores tres fases y que define el desarrollo realizado, son las pruebas de funcionalidad realizadas por el usuario. Para ello, la metodología XP propone el formato de la Tabla 14 para validación y aceptación por parte del usuario.

Tabla 14. Formato diligenciamiento Pruebas Funcionales

PRUEBA FUNCIONAL	
Número de Prueba:	**Número de Historia de Usuario:**
Nombre de la prueba:	
Descripción:	
Condición de ejecución:	
Entrada:	
Resultado Esperado:	
Evaluación de la prueba:	

Definiciones:

- Número de pruebas: Identifica la numeración de la prueba a ejecutar.
- Número de Historia de usuario: Define la historia de usuario que será evaluada en la prueba.
- Nombre de la prueba: Identificación general de la prueba.
- Descripción: Relato detallado del proceso que se deberá validar en el proceso de prueba.
- Condición de ejecución: Parámetros condicionales que se deben cumplir para el inicio de la prueba.
- Entrada: Datos iniciales de ingreso.
- Resultado esperado: Validaciones de los procesos que se deberían cumplir en la prueba realizadas.
- Evaluación de prueba: Resultado positivo o negativo de la prueba respecto a los resultados esperados.

- Para obtener mayor información respecto a las pruebas realizadas, dirigirse al Anexo F – Pruebas.

Capítulo 4

Resultados y Discusión

4.1 Introducción

Entendiendo los objetivos planeados en este proyecto, se ha desarrollado a primera versión del producto de software denominado: *Sistema de Control de Versiones para el Desarrollo de Software Seguro*, en el cual, se ha dado respuesta a las necesidades y especificaciones del cliente garantizando el proceso y gestión documental de cada uno de los artefactos de las fases de desarrollo de software seguro.

El Sistema permitirá al administrador o gerente del proyecto, llevar control del proceso evolutivo de tal manera que, pueda cumplir con su propósito de dar solución al cliente sin dejar a un lado la implementación de proceso de seguridad que garantizará la calidad del software.

4.2 Especificaciones de la aplicación.

La aplicación cumple con el objetivo propuesto, es amigable con un diseño intuitivo, enfocado al tema propuesto, entre sus características encontramos:

- Lending page: También conocida como la página de inicio y parte pública, contiene las secciones esenciales para enfocar al usuario en cuanto a lo encontrado en la aplicación.
- La interfaz de usuario: Medio establecido de comunicación entre el usuario y la aplicación. En esta aplicación la interfaz de usuario está destinada a entregar información acerca de los procesos y herramientas de control, a través de lo observado en la pantalla.

- Color: Se utiliza el color azul para realzar, resaltar o reiterar información mostrada como el único medio de transmitir información, además de transmitir tranquilidad y seriedad.
- Elementos de pantalla**:** Se definieron menú lateral izquierdo de navegación de aplicación, menú superior de idioma y datos de usuario, un área de trabajo con estándar de título, campo de búsqueda y listado de registros.
- Diseño interpretativo**:** Para ayudar a los usuarios que no pueden ver el contexto de un objeto en la pantalla, se asigna a cada objeto una etiqueta descriptiva y única de título que se activa al pasar el mouse sobre ella.
- Tamaño de fuente**:** El tamaño del texto y de los gráficos permite a los usuarios cambiar el tamaño de los objetos de la pantalla y utilizar las medidas del sistema para predefinir las preferencias del usuario.
- Adaptabilidad**:** Las interfaces están diseñadas para adaptarse a los diferentes tipos de pantalla en los diferentes dispositivos de visualización.
- Interfaz general: Es lo suficientemente flexible como para ajustarse a las necesidades y preferencias del usuario.
- Formularios: Con título, encabezado descriptivo de funcionalidad y con uniformidad en todas sus presentaciones.

4.3 Resultados - Características del software

Mediante la conexión a internet, el usuario tendrá la posibilidad de acceder desde cualquier equipo y en cualquier momento con un usuario y contraseña registrado previamente en la base de datos cuyo sistema gestor relacional utilizado será Mysql (Data Base Master System).

Funcionalmente, el sistema permitirá el ingreso de usuarios previamente creados en el sistema, quienes tendrán accesos a los diferentes menús de administración (Empresas, Usuarios, Proyectos, Metodologías, fases, versiones) acorde a los perfiles asignados en su creación.

El sistema será cargado inicialmente con Metodologías para el desarrollo de software seguro genéricas al igual que sus respectivas fases, deberán ser creados los proyectos, usuarios administradores y auxiliares que operarán el sistema, estos primeros tendrán la posibilidad de modelar una nueva técnica de desarrollo seguro según la disposición del usuario, de esta manera el sistema permitirá el versionamiento de los artefactos definidos para cada fase utilizada en los proyectos de desarrollo de software para los cuales se implemente.

El proceso de generación de versiones se llevará a cabo respecto a documentos base que serán cargados para cada fase que se asigne a la metodología y esta, podrá almacenar más de un documento base de ser requerido por el usuario, lo que permite el dinamismo en cuanto a la cantidad de artefactos que se requieren versionar en cada una de las etapas. El usuario descargará el documento y subirá desde la Versión 1 hasta el versionamiento que

se encuentre al momento de ser aprobado por el usuario administrador que puede ser el gerente del proyecto o, su jefe inmediato.

El Sistema llevará el control de las versiones y servirá de repositorio de los documentos, ayudando con la gestión documental del proyecto.

La estructura y diseño que se trabajará en el desarrollo del software permite la modificación de las metodologías utilizadas por la empresa, podrá dejar la actual a modo de consulta de tal manera que quede el registro de los procesos ejecutados con ella y, tendrá la posibilidad de crear una nueva en caso de ser requerido.

Por tratarse de un ambiente WEB, se requiere que el usuario cuente con acceso a internet continuo para el ingreso y navegación del sistema.

Los navegadores que permiten el correcto funcionamiento del repositorio son:

- Mozilla Firefox
- Google Chrome

El Sistema de Control de Versiones para el Desarrollo de Software Seguro, cuenta con las siguientes funciones principales:

- Uso Multi-empresas: Entendiendo la posibilidad de uso del sistema por diferentes empresas, el sistema permite la creación de diferentes entidades garantizando a privacidad de su información y, permitiendo el ingreso de múltiples usuarios con diferentes perfiles en un mismo momento.
- Gestión y Administración de Empresas, Metodologías, Fases, Proyectos, Usuarios, Versiones: Se establecen los permisos administrativos a los usuarios creados dependiendo de los tres niveles:

  - Usuarios Súper administradores: Usuario con roles que permiten la creación de empresas y usuarios de segundo nivel encargados de sus administraciones. Podrá cargar en el sistema las Metodologías para el desarrollo de Software Seguro catalogadas como genéricas y que hacen referencia a las técnicas encontradas en el mercado. Tendrá acceso a la activación o inactivación de empresas o usuarios creados.
  - Usuarios Administradores: Creados y asignados a una única empresa con el fin de Gestionar los procesos relacionados y la información de su entidad, incluye la creación y asignación de metodologías a proyectos desarrollados en su empresa, al igual que la creación de usuarios para ella misma.
  - Usuario Auxiliar: Creado y asignado con la función de generar las nuevas versiones de los documentos en cada fase de la metodología aplicada al proyecto.
- Repositorio: El Sistema contará con la funcionalidad de almacenamiento de las diferentes versiones de artefactos de cada fase ejecutada; esta información podrá ser consultada en cualquier momento para tener de esta manera cada versión controlada y confiable.
- LACMI: Funcionalidad administrativa de los usuarios en los diferentes niveles – Listar, Adicionar, Consultar, Modificar, Inactivar.
- Control de Versiones: Podrá llevarse el proceso de cambios y actualizaciones de los diferentes documentos, almacenado en versiones hasta llegar a una aprobación.

- Adaptabilidad a diferentes metodologías: Se permite la asignación de metodologías genéricas creados por el usuario Súper Administradores en cualquier empresa y proyecto, garantizando la seguridad de los documentos.
- Creación de metodologías propias: Basadas en las fases y metodologías genéricas, podrán ser creadas metodologías privadas con fases tomadas de las genéricas ya establecidas y, agregar nuevas de ser requerido.
- Seguimiento de proyectos y Gestión documental: Los proyectos ejecutados podrán ser consultados con el fin de conservar el histórico de lo realizado en determinado momento.

Los perfiles de seguridad definidos, identifican los siguientes accesos a los usuarios:

- Usuario Súper Administrador: Listar, Adicionar, Consultar, Modificar, Inactivar
    - Empresas
    - Metodologías (Públicas o privadas si lo requiere)
    - Fases (Públicas o privadas si lo requiere)
    - Proyectos (Solo asignados en su nivel)
    - Versiones y documentos (Solo asignados a los proyectos privados de su nivel).
- Usuario Administrador: Listar, Adicionar, Consultar, Modificar, Inactivar
    - Metodologías Privadas
    - Fases
    - Proyectos solo asignados a su empresa
    - Versiones y documentos solo asignados a los proyectos de su empresa.

- Usuario Auxiliar: Listar, Adicionar, Consultar, Modificar
    - Proyectos solo asignados a su empresa – Solo función Listar - Consultar
    - Versiones y documentos solo asignados a los proyectos de su empresa.

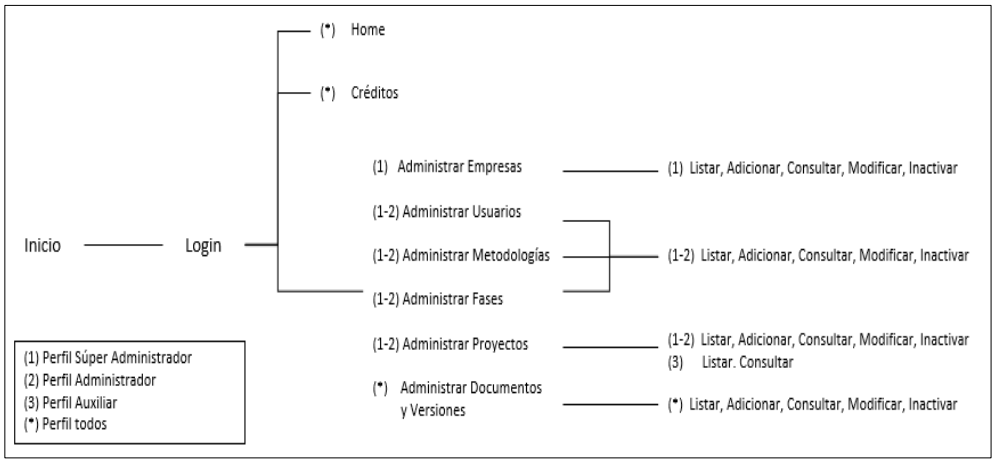


Figura 14. Ruta de navegación y accesos por Perfil de usuario

Como definición en el proceso de versionamiento de documentos, se ha definido la asignación automática de la enumeración por parte del sistema de control de versiones, de esta manera se controla el estándar del versionador. Esta asignación se llevará a cabo mediante números enteros partiendo de la Version1 (V1), hasta la versión que se aprobada. Cada fase contará con su propio contador y asignador de número de versión iniciando en V1. Así mismo, el sistema contará con la visualización del seguimiento en tiempo real de los proyectos que se están trabajando en el momento, con el fin de validar el status de cada fase y el avance total del proceso.

4.3.1 Interfaz de Usuario

Se establecen diferentes interfaces de usuario con el fin de permitir la correcta navegación del Sistema.

- Interfaz de Ingreso: La interfaz de usuario permite la validación de datos y credenciales, no se permite el ingreso de usuario sin previa creación (Figura 15).

Figura 15. Interfaz de usuario – ingreso

- Interfaz de Ingreso – Home: Los usuarios tendrán cuatro diferentes vistas con la información básica del Desarrollo de Software Seguro, como se ve en la Tabla 15.

Tabla 15. Datos informativos Sistema Control de Versiones

	Versionador
	S-SDLC
	Servicios
	Contacto
	Creditos

- Interfaz de Menú: Los accesos al menú de administración para las diferentes opciones, se dividen de acuerdo al usuario que ingresa.

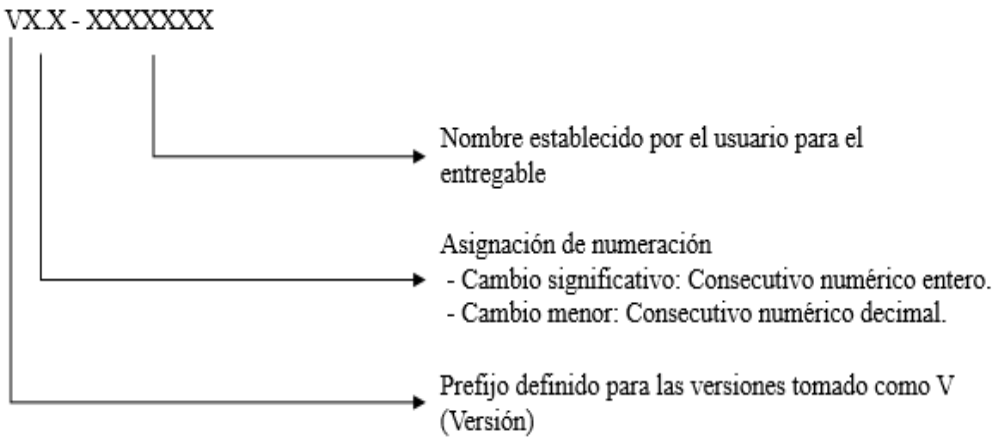
Para un mejor uso del producto, se sugiere validar el proceso funcional que se encuentra en el Anexo H – Manual de Usuario y, y los process de instalación en Anexo G – Manual de Instalación.

4.4 Discusión

Como parte de un proceso de mejora u optimización, el Sistema Control de Versiones podría contar con asignación dinámica de versiones, de tal manera que el usuario pueda definir:

4.4.1 Codificación

La estandarización en el código de versión tendría la siguiente estructura:



Como ejemplo tenemos la Tabla 16:

V1.0 – Casos de Abuso	Cambio Significativo
V1.1 – Casos de Abuso	Cambio menor

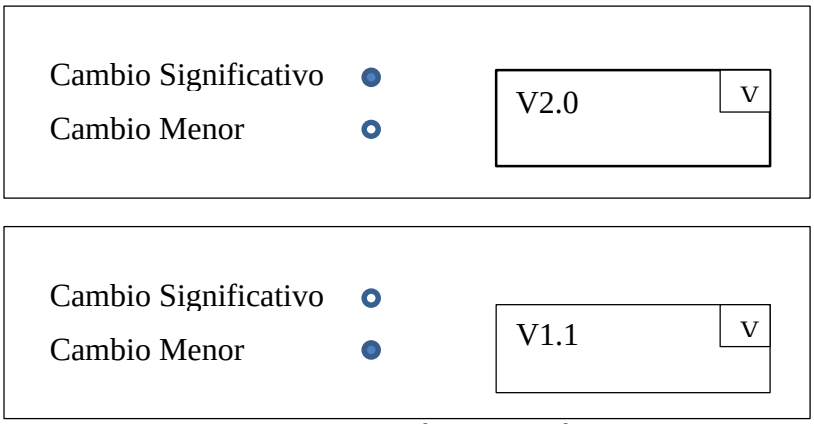


Tabla 16. Prototipo de asignación de versión para documentos

4.4.2 Mejoras a nivel de seguridad

Como punto de mejoramiento a nivel de seguridad, se presentan los siguientes items:

4.4.2.1 Administración de Roles y perfiles

Creación de proceso de "Administración de Roles y Perfiles", con las características similares a las encontradas en las demás administraciones de la Versión 1 del Sistema de Control de Versiones para el Desarrollo de Software Seguro, esta Administración, solo podrá ser gestionada por el Súper administrador del Sistema, quien otorgará los diferentes perfiles a los usuarios.

4.4.2.2 Creación de Bitácoras de seguridad

Para controlar los diferentes ingresos, modificaciones o procesos ejecutados en el sistema, podrá ser construida una bitácora en la cual, se registren la fecha y hora de:

- Intentos fallidos de ingreso.

- Ultimo ingreso al sistema
- Cambio de clave
- Transacción o procesos ejecutados

4.4.3 Repositorio y Versiones

La flexibilidad del sistema podrá contemplar el control de versiones de entregables relacionados a las fases de la metodología de desarrollo y ciclo de vida establecido para el proyecto (metodología de desarrollo tradicional), de esta manera, al ser asignada la metodología a un proyecto, y apalancado en la funcionalidad actual en la que es posible subir varios documentos o plantillas en una fase, se subirán ambos entregables, diferenciadas por un icono de seguridad. Ejemplo:

- Metodología Propia
    - Fase de Requerimiento
        - Requerimientos Funcionales y no funcionales del usuario
        -  Requerimientos de seguridad

Capítulo 5

Lista de referencias

Warren Axelrod, C.(2013), *Engineering safe and secure software systems*, Artech House, Boston, Londres

Barco, V. (1989). *Ley 527 de 1999 Nivel Naciona.* Retrieved from Régimen Legal de Bogota DC: http://www.alcaldiabogota.gov.co/sisjur/normas/Norma1.jsp?i=4276

Bootstrap. (2016). Retrieved from http://getbootstrap.com/

Cigital, Inc. (2016). *SSDLC 101: What Is the Secure Software Development Life Cycle?* Retrieved from Cigital: https://www.cigital.com/blog/what-is-the-secure-software-development-lifecycle/

Colciencias.gov. (2009). *GRINDTIC*. Retrieved from http://scienti.colciencias.gov.co:8080/gruplac/jsp/visualiza/visualizagr.jsp?nro=00000000009603

Common Criteria. (2012). *Evaluation, Part 1: Introduction and general model. V3.* Retrieved from Common Criteria for Information Technology Security: https://www.commoncriteriaportal.org/files/ccfiles/CCPART1V3.1R4.pdf

Consejo Nacional de Política Económica y Social. (2016). *Política Nacional de Seguridad Digital.* Retrieved from Documento Conpes: https://colaboracion.dnp.gov.co/CDT/Conpes/Econ%C3%B3micos/3854.pdf

Curello Fabio. (2013). *Desarrollo seguro usando OWAS.* Retrieved from The Owas Fundation: https://www.owasp.org/index.php/Category:OWASP_CLASP_Project

Fernado Catoira. (2016). *Detectando Vulnerabilidades de Software mediante Fuzzing.* Retrieved from Welivesecurity - ESET: http://www.welivesecurity.com/la-es/2012/10/31/detectando-vulnerabilidades-softwar-mediante-fuzzing/

Free Software Fundation. (2016). *Licencia Pública General de GNU.* Retrieved from El Sistema Operativo GNU: https://www.gnu.org/licenses/licenses.es.html

Gandini I., Isaza A, Delgado A. (n.d.). *Ley de Delitos Informáticos.* Retrieved from Delta Asesores: http://www.deltaasesores.com/articulos/autores-invitados/otros/3576-ley-de-delitos-informaticos-en-colombia

Gobierto de España.Ministerio de Hacienda y Administraciòn Publica. (2016). *Metodologia de análisis y Gestion de Riesgos de los Sistemas de Información.*

Retrieved from Margerit : http://administracionelectronica.gob.es/pae_Home/pae_Documentacion/pae_Metodolog/pae_Magerit.html#.V5l-JfkrKM8

Group, P. (2001-2006). *PHP*. Retrieved from PHP: http://www.php.net/

IEEE. (2002).

Correctness by construction: developing a commercial secure system. Retrieved from IEEEXplore Digital Library: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=976937&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel5%2F52%2F21072%2F00976937.pdf%3Farnumber%3D976937

ISO. (2008). *Information technology -- Security techniques -- Systems Security Engineering -- Capability Maturity Model® (SSE-CMM®).* Retrieved from ISO: http://www.iso.org/iso/catalogue_detail.htm?csnumber=44716

jQuery Fundation. (2016). *JQuery User Interface*. Retrieved from jQuery: https://jqueryui.com/

jQuery Fundation. (2016). *what is jQuery?* Retrieved from jQuery Write less, do more: https://jquery.com/

Laravel. (2016). *Hash*. Retrieved from Laravel: http://laraveles.com/docs/5.1/hashing

Letelier, P., & Penadés, M. C. (2006). *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* Retrieved from http://www.cyta.com.ar/ta0502/v5n2a1.htm

Markus Erb. (n.d.). *Amenazas y Vulnerabiliades*. Retrieved from Gestión de Riesgo en la Seguridad Informática: https://protejete.wordpress.com/gdr_principal/amenazas_vulnerabilidades/

Mc Graw, G., & Jhon, V. (2006). *Building Secure Software: How to Avoid Security Problems the Right Way.* Mexico DF: Addison-Wesley.

Microsoft. (2016). *Análisis de modelo de amenazas*. Retrieved from https://msdn.microsoft.com/es-es/library/aa561499.aspx

Microsoft. (2016). *Security Development Lifecycle.* Retrieved from Microsoft: https://www.microsoft.com/en-us/sdl/

Ministerio de Educación, Cultura y Deporte. Gobierno de España. (2012). *Vulnerabilidades de un Sistema Informatico*. Retrieved from Introducción a la

Seguridad Informática: http://recursostic.educacion.es/observatorio/web/gl/software/software-general/1040-introduccion-a-la-seguridad-informatica?start=3

OpenSAMM. (n.d.). *Software Assurance Maturity Model*. Retrieved from OpenSAMM: http://www.opensamm.org/

Oracle. (2016). *MySQL*. Retrieved from MySQL: https://www.mysql.com/

OWASP. (2016, Mayo 16). *OWASP CLASP Project*. Retrieved from OWASP: https://www.owasp.org/index.php/OWASP_CLASP_Project

PHP Corporation. (2016). *PHP*. Retrieved from PHP: http://www.php.net/

SANS Institute. (2008). Retrieved from SANS Software Security: http://software-security.sans.org/

SANS Institute. (2008). Retrieved from SANS Software Security.

Albasanz.c (s.f). Manual XML. http://www.mundolinux.info/que-es-xml.htm

Brea, Orlando F. (2005). WSDL y UDDI. http://www.desarrolloweb.com/articulos/1857.php

Oasis.(2016).OASIS Web Services Security (WSS) TC. https://www.oasis-open.org/committees/tc_home.php?wg_abbrevtiwss

Cabrera, Luis F. (2015). WEB Services Atomic Transactions. V1. http://specs.xmlsoap.org/ws/2004/10/wsat/wsat.pdf

Microsoft. (s.f). Análisis de Modelo de Amenaza. shttps://msdn.microsoft.com/es-es/library/aa561499.aspx

Bytes & Chips. (2014). Check list de requisitos no funcionales. registro.pse.com.co

Universidad Unión Bolivariana.2006.Ingenieria de software. Programación Extrema XP, Recuperado de http://downloads.mysql.com/docs/workbench-en.pdf

Gary McGraw, (2009), Software Security Building Security in http://www.swsec.com/resources/

Normas APA, (2016), Normas APA actualizadas 2016, http://normasapa.com/

Patricio Letelier, Mª Carmen Penadés. Universidad Politécnica de Valencia (UPV).(2006). Recuperado de http://www.cyta.com.ar/ta0502/v5n2a1.htm

Instituto Nacional de Tecnologías de la comunicación. Inteco. Ingeniería del Software seguro: Metodologías y ciclos de vida. Recuperado de http://es.slideshare.net/jesws1/guia-para-desarrollo-de-software-seguro

Brito Abundis Carlos J., (ReCIBE, año2 núm. 3, 2013), Metodologías para el desarrollo De software seguro. http://recibe.cucei.udg.mx/revista/es/vol2-no3/pdf/computacion05.pdf?ver=24062013

ISO/IEC 15408-1:2009. Information technology – Security techniques - Evaluation criteria for IT security -- Part 1: Introduction and general model. https://www.iso.org/obp/ui/#iso:std:iso-iec:15408:-1:ed-3:v2:en

Official website of the Department of Homeland Security.Building Security In Setting a higher standard for software assurance. https://buildsecurityin.us-cert.gov/resources/websites/clasp

Chandra Pravir. Software Assurance Maturity Model. http://www.opensamm.org
Documento Compes. Politica Nacional de Seguriad Nacional. (2016). http://www.mintic.gov.co/portal/604/articles-14481_recurso_1.pdf

Otero Gutierrez D.(2011).Desarrollo de una Aplicación WEB para control de versiones de software.
http://e-archivo.uc3m.es/bitstream/handle/10016/11936/PFC-David%20Otero%20Gutierrez.pdf?sequence=1&isAllowed=y

Tello- Leal Edgar, Sosa Clauda, Tello- Leal Diego.(2012)
Revision de los Sistemas Control de Versiones utilizados en el desarrollo de Software. http://web.usbmed.edu.co/usbmed/fing/v3n1/v3n1a9.pdf

Dinero.(2016). El 2015 fue un año de “Altas y Bajas” para la seguridad informática. http://www.dinero.com/pais/articulo/informe-certicamara-sobre-seguridad-informatica-colombia-para-2016/217635

El Tiempo.(2015). Colombia es el tercer país de la región con mas ataques informáticos. http://www.eltiempo.com/tecnosfera/novedades-tecnologia/ataques-ciberneticos-en-latinoamerica/16383752

Régimen Legal de Bogota DC(1999). Decreto 1360 de 1999 Nivel Nacional. http://www.alcaldiabogota.gov.co/sisjur/normas/Norma1.jsp?i=10575

Gandini I., Isaza A, Delgado A. Ley de Delitos Informáticos (Delta asesores)(s,f). http://www.deltaasesores.com/articulos/autores-invitados/otros/3576-ley-de-delitos-informaticos-en-colombia.

Rodriguez V., Fernandez L., XV Congreso internacional de Ingeniería de Proyectos (2011). http://www.aeipro.com/files/congresos/2011huesca/CIIP11_2390_2405.3423.pdf

MINTIC Colombia (2016). Conpes de Seguridad Digital está acorde con estándares de la industria TI. http://www.mintic.gov.co/portal/604/w3-article-15463.html

El Tiempo.Ataques cibernéticos en Latinoamerica.(2016). http://www.eltiempo.com/tecnosfera/novedades-tecnologia/ataques-ciberneticos-en-latinoamerica/16383752.

Dinero. Seguridad Informática en Colombia. (2015). http://www.dinero.com/pais/articulo/informe-certicamara-sobre-seguridad-informatica-colombia-para-2016/217635

Recibe, Revista Electrónica. (2.013). Metodologías para el desarrollo de software seguro. http://recibe.cucei.udg.mx/revista/es/vol2-no3/computacion05.html

Fondo Nacional de Garantías (FNG). (2008). Ley de Hábeas Data Ley 1266 de 2008. https://www.fng.gov.co/ES/Documentos%20%20Proteccin%20de%20Datos%20Personales/Manual%20Habeas%20Data.pdf

Enter.co. (s,f,).El 68% de las empresas en latinoamérica sufrió ataques informáticos. http://www.enter.co/especiales/enterprise/el-68-de-las-empesas-en-latam-sufrieron-ataques-informaticos/