METODOLOGÍA AGIL PARA EL DESARROLLO DE SOFTWARE WEB BASADA EN SCRUM Y PATRONES DE SEGURIDAD OWASP QUE FORTALECEN LA SEGURIDAD DE LA INFORMACIÓN COMO ATRIBUTO DE CALIDAD DEL PRODUCTO FINAL

METODOLOGÍA AGIL PARA EL DESARROLLO DE SOFTWARE WEB BASADA EN SCRUM Y PATRONES DE SEGURIDAD OWASP QUE FORTALECEN LA SEGURIDAD DE LA INFORMACIÓN COMO ATRIBUTO DE CALIDAD DEL PRODUCTO FINAL

SCOWP

Augusto César De Arco Chiquillo

Directora

Msc. Adriana María Iglesias Solano

Universidad Simón Bolívar

Maestría en Ingeniería de Sistemas y Computación

Barranquilla, Octubre de 2018

Dedico los resultados de este trabajo de investigación a:

***Augusto e Inés**, mis padres y ejemplos a seguir*

***Rita y Rafael**, mis hermanos y compañeros de lucha*

***Sergio, Isabel y Analía**, mis sobrinos del alma*

***Jessica**, mi esposa y coequipera de aventuras*

***Julián Felipe**, mi hijo y motivación a ser cada día un mejor guía en su vida*

Por ellos todo, sin ellos nada

Contenido

Resumen 1
Abstract 2
Introducción 1
1. Planteamiento del problema y Justificación 2
2. Objetivos 5
2.1. Objetivo general 5
2.2. Objetivos específicos 6
3. Metodología de la investigación 6
4. Hipótesis del trabajo 7
5. Revisión de la literatura 8
5.1. Panorama económico del sector software 8
5.2. Metodologías de desarrollo de software 13
5.2.1. Metodologías tradicionales 13
5.2.2. Metodologías ágiles 19
5.2.3. Comparativo de las metodologías ágiles con las tradicionales 30
5.3. Elicitación de necesidades 32
5.4. La seguridad informática 33
5.5. La seguridad en aplicaciones web 35
5.6. Rol de las pruebas en las metodologías de desarrollo de software 39
5.7. Open Web Application Security Project (OWASP) 42
5.8. Ciclos de vida de desarrollos de software seguro 49
6. Caracterización de la metodología propuesta. 51
6.1. ¿Por qué SCRUM? 51
6.2. ¿Por qué OWASP? 55
7. Metodología propuesta (SCOWP) 56
7.1. Elementos de la metodología propuesta 59
7.1.1. Dueño del producto 59
6.1.2. Guía metodológico 59
6.1.3. Equipo de desarrollo 61
6.1.4. Equipo de pruebas 62
7.2. Eventos de la metodología 63
7.2.1. Elicitación de necesidades mediante Design Thinking. 63
7.2.2. Reunión de Elaboración/Revisión de documentos OWASP 65

7.2.3. Reunión de análisis de riesgos OWASP 66
7.2.4. Reunión de análisis de necesidades funcionales 67
7.2.5. Reunión de análisis de necesidades no funcionales 67
7.2.6. Reunión de Planificación del Sprint 68
7.2.7. El Sprint 69
7.2.8. Reunión diaria 70
7.2.9. Ejecución de pruebas 70
7.2.10. Revisión del Sprint 71
7.2.11. Retrospectiva del Sprint 72
7.3. Artefactos de la metodología 73
7.3.1. Producto "Terminado" versus incremento 73
7.3.2. Lista de producto 74
7.3.3. Lista de pendientes 75
7.3.4. Matriz OWASP de Riesgos, Vulnerabilidades y Defensas 76
7.3.5. Árbol de utilidad 86
7.3.6. Escenarios de calidad 86
7.3.7. Casos de pruebas OWASP-RVD 87
7.3.8. Casos de pruebas funcionales 87
7.3.9. Formato de resultados de pruebas/validación 88
7.3.10. Aceptación de producto "Terminado" 88
8. Prueba piloto de la metodología SCOWP 89
8.1. Requerimientos del cliente 89
8.2. Resultados de la prueba piloto 91
9. Conclusiones 93
10. Referencias 96
11. Anexos 101
11.1. Anexos Documentales 101
11.1.1. Lista de producto 101
11.1.2. Lista de pendiente 102
11.1.3. Matriz OWASP-RVD 103
11.1.4. Árbol de Utilidad 104
11.1.5. Escenarios de calidad 105
11.1.6. Casos de Pruebas OWASP-RVD 106
11.1.7. Casos de pruebas funcionales 107
11.1.8. Formato de resultados de pruebas/validación 108

11.1.9. Aceptación de producto “Terminado” 109
11.2. Anexos prueba piloto 110
10.2.1. Lista de Producto 110
10.2.2. Lista de Pendiente 111
10.2.3. Árbol de Utilidad 112
10.2.4. Escenarios de calidad 113
10.2.5. Casos de pruebas funcionales 114
10.2.6. Casos de pruebas OWASP-RVD 117
10.2.7. .Formato de resultado de pruebas/validación 121
10.2.8. Aceptación de producto “Terminado” 124

Lista de tablas

Tabla 1. Valores y principios del manifiesto ágil 20

Tabla 2. Comparación de Metodologías ágiles vs. Metodologías tradicionales 31

Tabla 3. Descripción de Riesgos OWASP Top 10-2017 43

Tabla 4. Descripción de Controles Proactivos OWASP 2018 46

Tabla 5. Fases y Prácticas de SDLC 50

Tabla 6. Ejemplo de información que debe contener la Matriz OWASP-RVD 77

Lista de Figuras

Figura 1. Ataques a aplicaciones web en Tercer trimestre de 2017 3

Figura 2. Distribución de empresas desarrolladoras de software en Colombia en 2018 9

Figura 3. Facturación por línea de negocio a nivel nacional en 2015. 10

Figura 4. Ingresos históricos de la industria del software 11

Figura 5. Exportaciones históricas de software (USD Millones) 12

Figura 6. Países donde se exporta el producto software hecho en Colombia 12

Figura 7. Modelo en cascada de desarrollo de software 16

Figura 8. Esquema SCRUM 25

Figura 9. Tablero Kanban 28

Figura 10. Familia de metodologías Crystal según número de miembros en el proyecto 29

Figura 11. Nivel de peligrosidad de los ataques provenientes de Internet 36

Figura 12. Ejemplos de implementación de Captcha 39

Figura 13. Evaluaciones estáticas y dinámicas en el proceso de desarrollo de software 40

Figura 14. Modelo en V de la evaluación de software 41

Figura 15. Costo de solución de vulnerabilidades v.s. etapa del software 50

Figura 16. Implementación de metodologías ágiles en VersionOne-Año 2015 53

Figura 17. Implementación de metodologías ágiles en VersionOne-Año 2016 54

Figura 18. Implementación de metodologías ágiles en VersionOne-Año 2017 54

Figura 19. Esquema de la Metodología SCOWP 58

Figura 20. Imagen de aplicativo piloto-Versión App 91

Resumen

En esta Tesis de maestría se abordó como objeto de estudio el bajo nivel de seguridad presentados en los desarrollos de aplicativos orientados a la web que permiten afectaciones importantes a la información. Se realizó un diagnóstico de la situación actual de las metodologías ágiles y estándares de seguridad de la información en proyectos de desarrollo de software, consiguiendo caracterizar los principales componentes de Scrum y Owasp para formular a través de un método descriptivo/propositivo una metodología ágil denominada SCOWP para el desarrollo de software orientado a la web que incluye dentro de sus etapas el cumplimiento de controles a vulnerabilidades y tiene en cuenta los atributos de calidad generados por necesidades no funcionales de seguridad. Se logró demostrar que es posible mejorar las técnicas de desarrollo de software web fortaleciendo los aspectos de seguridad de la información como un atributo de calidad que valoriza el producto final.

Palabras Clave: Metodologías ágiles, vulnerabilidades de software, seguridad de la información, riesgos informáticos, SCOWP

Abstract

In this thesis document, the low level of security presented in the development of web-oriented applications that allow significant impact on the information was addressed as an object of study. A diagnosis was made of the current situation of agile methodologies and information security standards in software development projects, managing to characterize the main components of Scrum and Owasp to formulate through a descriptive / proactive method an agile methodology called SCOWP for the development of web-oriented software that includes, within its stages, compliance with vulnerability controls and takes into account the quality attributes generated by non-functional security needs. It was demonstrated that it is possible to improve web software development techniques by strengthening the information security aspects as an attribute of quality that values the final product.

Keywords: Agile methodologies, software vulnerabilities, information security, computer risks, SCOWP

Introducción

En esta Tesis de Maestría de Ingeniería de sistemas y computación se aborda el tema de las amenazas que presentan los desarrollos de aplicativos orientados a la web y cómo pueden reducirse los riesgos con la aplicación de controles teniendo conocimiento de ellos desde la fase de análisis y desarrollo del software, sin dejar la carga de responsabilidad exclusivamente a la etapa de pruebas.

El propósito fundamental es crear una metodología ágil de desarrollo de software que implemente controles que fortalezcan la seguridad de la información como atributo de calidad asociado al producto final, basándose en la metodología ágil Scrum y las defensas determinadas por OWASP para las vulnerabilidades detectadas en los aplicativos orientados a la web.

Durante la etapa de diagnóstico de la situación actual se encontró información estadística importante sobre los ataques informáticos a través de internet, muchos de ellos atribuibles a las vulnerabilidades que presentan los aplicativos por un deficiente análisis de los riesgos en las etapas de desarrollo y una deficiente etapa de pruebas, sobre todo en las metodologías ágiles que centran sus esfuerzos en entregas rápidas con el cumplimiento de las necesidades funcionales, dejando de lado aspectos no funcionales como la seguridad. Si bien en la actualidad existe una metodología orientada al desarrollo seguro de software web propuesta por Microsoft, propongo la formulación teniendo en cuenta estudios detallados sobre los riesgos frecuentes sobre los aplicativos expuestos al internet. El resultado final es una metodología ágil para el desarrollo de software orientado a la web que fortalece el análisis de los riesgos de

ataques desde internet implementando buenas prácticas de seguridad asociadas a las necesidades no funcionales y al análisis de patrones entregado por OWASP. El manejo de las necesidades funcionales se trata someramente porque las metodologías existentes las abordan ampliamente y el objetivo principal planteado se enfoca en el fortalecimiento de la seguridad como atributo de calidad del software como producto final

1. Planteamiento del problema y Justificación

El desarrollo de software se rige desde el punto de vista comercial por los requerimientos del cliente para cubrir una necesidad puntual dentro de un proceso. Cuando el software pasa a ambiente productivo, el cliente evalúa funcionalmente el programa de computador y centra su atención en la satisfacción de las necesidades que fueron consignadas en el contrato establecido con la empresa desarrolladora. Sin embargo, en muchas ocasiones ni el desarrollador ni el cliente tienen en cuenta aspectos de seguridad que atenten contra la integridad de la información y del programa mismo.

La seguridad en el software es un componente que debe implementar el desarrollador porque es quien debe prever y analizar los escenarios de vulnerabilidad del producto que desarrolla dentro de su obligatoriedad de entregar un producto de calidad.

El escenario de ejecución de las aplicaciones web no tiene límite, una vez la aplicación está en productivo todo acceso es posible desde cualquier punto geográfico y los ciberdelincuentes se encuentran distribuidos en todo el planeta. En Colombia

durante el periodo de Agosto 2016 a Agosto 2017 se presentaron 198 millones de ataques informáticos que generaron pérdidas cercanas a los 6.200 millones de dólares al país. (El Tiempo, 2017)

Según el estudio de Adasoft (2017) para el año 2021 se estima que las personas dedicadas al cibercriminal triplicarán a los dedicados a la ciberseguridad por lo que es importante que quienes trabajan en el campo de la informática tengan conciencia que deben prevenir desde sus labores y tratar de reducir los riesgos.

Los estudios realizados por Digital Media Technologies sobre ataques en internet indican que las aplicaciones web son el blanco más apetecido cada vez más. Para el 2016, los ataques a nivel global aumentaron en un 30% solo entre el segundo y tercer trimestre, cifra que coincide con el tercer trimestre de 2017. Para el tercer trimestre de 2017 los vectores de ataque a aplicaciones web fueron Injection SQL y LFI (Inclusión por archivos locales), alcanzando en conjunto un 85% de los casos. (Dimtec, 2018).

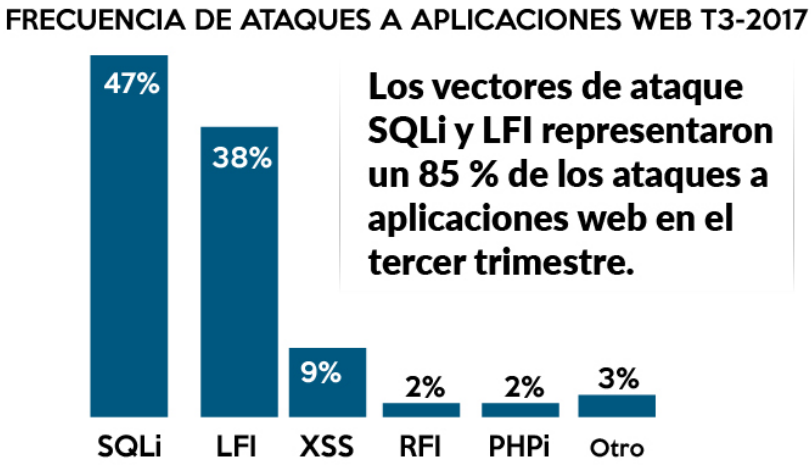


Figura 1. Ataques a aplicaciones web en Tercer trimestre de 2017

Fuente: (Dimtec, 2018)

El panorama que se visualiza amerita adoptar medidas para reducir los riesgos de ataques en las aplicaciones web pues como se evidencia a pesar de los esfuerzos de los profesionales de la informática y sectores de TI, los casos donde se afecta la información cada día aumentan.

El problema abordado para analizar en este estudio es en muchos casos, el bajo nivel de seguridad que se presenta en los desarrollos de software web que permiten afectaciones importantes a la información.

El conocimiento de los riesgos, la inclusión de medidas preventivas y establecimiento de controles dentro del proceso de desarrollo del software web pueden contribuir a mejorar la seguridad para la prevención de los ataques provenientes del ciberespacio.

Pregunta de investigación:

¿Es posible crear una metodología ágil para desarrollo de software web que contrarreste las vulnerabilidades más reconocidas para preservar la seguridad de la información?

Justificación

Para las organizaciones y empresas desarrolladoras de software orientado a la web la ocurrencia cada vez más creciente de ataques desde es un problema que impone la imperiosa necesidad de encontrar soluciones inmediatas toda vez que la información de las organizaciones que depositan su confianza en éstos aplicativos se encuentran en grave riesgo. Una de las soluciones desde el campo del desarrollo de software

orientado a la web es la de contar con una metodología ágil que contrarreste las vulnerabilidades existentes más conocidas para preservar la seguridad de la información. La situación es cada vez más preocupante porque si bien se implementan controles, los ataques presentan tasas de crecimiento de 30% de un trimestre a otro y las aplicaciones web siguen siendo uno de los objetivos de ataque más apetecidos por la ciber delincuencia. Este trabajo final de maestría propone como resultado de investigación a **SCOWP**, una metodología ágil basada en Scrum que incluye aspectos y patrones de seguridad OWASP en el proceso de desarrollo de software orientado a la web como atributo de calidad del producto final. **SCOWP** está diseñada para reducir el éxito de ataques informáticos y se ofrece para empresas desarrolladoras de software y unidades de TI de las organizaciones como una valiosa herramienta para afrontar el reto de fortalecer sus aplicativos y preservar la seguridad, integridad, confiabilidad y confidencialidad de su información.

2. Objetivos

2.1. Objetivo general

Crear una metodología ágil de desarrollo de software orientado a la web basada en el marco de trabajo SCRUM que incluya aspectos y patrones OWASP de seguridad de la información como atributo de calidad del producto final.

2.2. Objetivos específicos

- Diagnosticar la situación actual del uso de las metodologías ágiles y estándares de seguridad de la información en proyectos de desarrollo de software
- Identificar los componentes de la metodología ágil caracterizada y del estándar de seguridad de la información empleando conceptos de Scrum y Owasp respectivamente.
- Definir los componentes estructurales de la metodología ágil orientado a la web para desarrollo de software seguro identificando eventos, artefactos y elementos humanos, que la conforman
- Desplegar la metodología ágil orientada a la web para desarrollo de software seguro en un proyecto piloto de acuerdo a las características de entradas definidas.

3. Metodología de la investigación

Para el trabajo de investigación sigue una metodología **Descriptiva/Propositiva** con un enfoque **Cualitativo** que permite el análisis de la situación actual de las metodologías de desarrollo de software y las condiciones de la seguridad de las aplicaciones en ambiente web, planteando como resultado una metodología de desarrollo de software ágil fortalecida con la implementación de patrones de seguridad probados que incrementan la seguridad de la información. Se llevarán a cabo las siguientes fases:

- Fase de diagnóstico: Se realiza un diagnóstico de la situación de seguridad de los desarrollos orientados a la arquitectura web, así como el uso de metodologías ágiles en éste tipo de proyectos.
- Fase de Identificación: Se seleccionan los componentes de trabajo y cómo se integrarán en la propuesta.
- Fase de Definición: Se define la relación de los componentes en la nueva metodología y cómo interactúan en el sistema detallando los roles, eventos y entregables propios de su dinámica.
- Fase de Validación: Implementación de un piloto de desarrollo de software web empleando la metodología propuesta y se analizarán los resultados obtenidos como validación de la hipótesis planteada.

4. Hipótesis del trabajo

Es factible mejorar las técnicas de desarrollo ágil de software orientado a la web basado en el marco de trabajo de SCRUM teniendo en cuenta casos de ataques y vulnerabilidades analizados por OWASP de manera que el producto final cumpla con los aspectos asociados a los atributos de calidad de seguridad

5. Revisión de la literatura

5.1. Panorama económico del sector software

Como fue analizada por Alfonso (2017, 2018) Colombia es un país que ha fortalecido la industria del software y presenta en este campo cifras importantes, con un crecimiento notorio y una participación del 1,6% del Producto Interno Bruto (PIB) en el 2017 y una meta de llegar a obtener un 5% para el 2025.

La implementación de programas gubernamentales como “Plan Vive Digital” y el programa de “Fortalecimiento de la Industria de Tecnologías de la Información” (FITI), entre otros, impulsaron el desarrollo del sector software a mediados de la década del 2000 (Portafolio, 2015). Otras medidas impositivas para la misma época como la exención del IVA aportó de manera significativa para el repunte del sector.

Adicionalmente, hasta el año 2017 hubo un estímulo importante para el sector del desarrollo de software, como lo plantean Aguilera y De las Salas (2014) se establecieron beneficios en el impuesto sobre la renta para quienes implementaran o intervinieran en proyectos calificados como de investigación y desarrollo tecnológico, entre ellos desarrollo de software, lo que de alguna manera apoyó directamente la competitividad del sector en mercados extranjeros.

En ese crecimiento es protagonista Fedesoft, entidad que agrupa a las empresas del sector del Software en Colombia que actualmente cuenta con cerca de 7.000 afiliados, distribuidos de la siguiente manera por categorización de empresas:

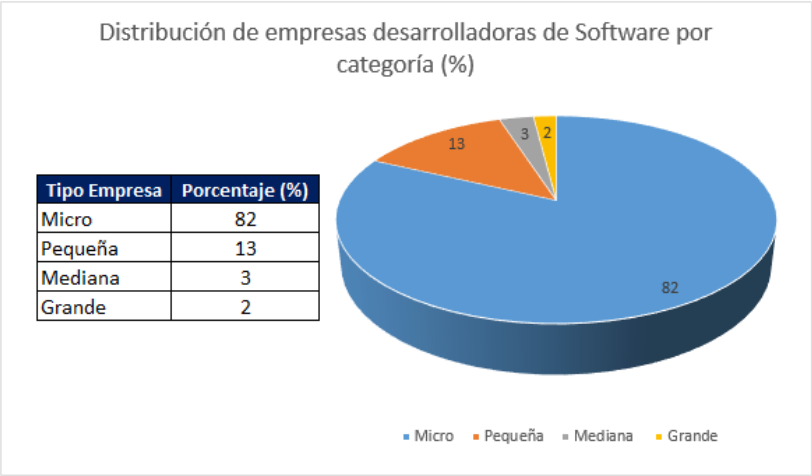


Figura 2. Distribución de empresas desarrolladoras de software en Colombia en 2018

Fuente: (Portafolio, 2018)

Portafolio (2018) afirma que “aunque solo el 2% de las empresas desarrolladoras de software se sitúan en la categoría de Grandes, éstas tienen una participación del 74% sobre el total de la facturación anual en éste renglón económico”.

Conforme lo registrado por Fedesoft (2015), el sector del desarrollo de software (desarrollo de aplicaciones, licenciamiento y software como servicio) en el renglón económico de las TIC es significativo, obteniendo para el 2015 una participación del 56,86% sobre lo facturado por Servicios de TI.

Software como servicio (SaaS)	27.09%
Desarrollo/fábrica de software	23.67%
Venta o licenciamiento de software	6.10%
Consultoría e implementación	4.65%
Servicios profesionales para TI	4.59%
Venta de hardware	3.47%
Mantenimiento o soporte de aplicaciones	3.23%
Servicios de conectividad	2.82%
Plataformas tecnológicas como servicio (PaaS)	2.77%
Integración de soluciones	2.54%
Seguridad informática	1.38%
Cloud computing (incluyen servicios integrados de SaaS, IaaS y PaaS)	1.07%
Data Center	1.04%
Mesa de ayuda/ Soporte infraestructura	0.85%
Testing de software	0.76%
Infraestructura como servicio (IaaS)	0.34%
Servicios de cableado	0.21%

Figura 3. Facturación por línea de negocio a nivel nacional en 2015.

Fuente: (Fedesoft, 2015)

De acuerdo al estudio de UNP (2015) entre el 2003 y el 2013 el mercado de software & TI en Colombia creció 5 veces su tamaño y gran parte de ello se debió a que el Gobierno estableció los “nuevos sectores basados en la innovación”, como una de las cinco locomotoras que debían jalonar el crecimiento del país en su plan de desarrollo. También contribuyó a la consecución de esta meta, la consolidación del acuerdo marco de precios para Equipos Tecnológicos y Periféricos en Colombia.

¿Cómo se refleja en los ingresos de las empresas de desarrollo de software el crecimiento del sector? El incremento que han tenido los ingresos de las empresas de desarrollo de Software en los últimos 5 años se ha duplicado. En el 2013 los ingresos de la industria fueron de $8,3 Billones y para el 2018 estaría cerrando en $16,1 Billones.

Año	Ingresos ($Billones)
2013	8,3
2014	9,3
2015	12,9
2016	13,5
2017	14,7
2018	16,1

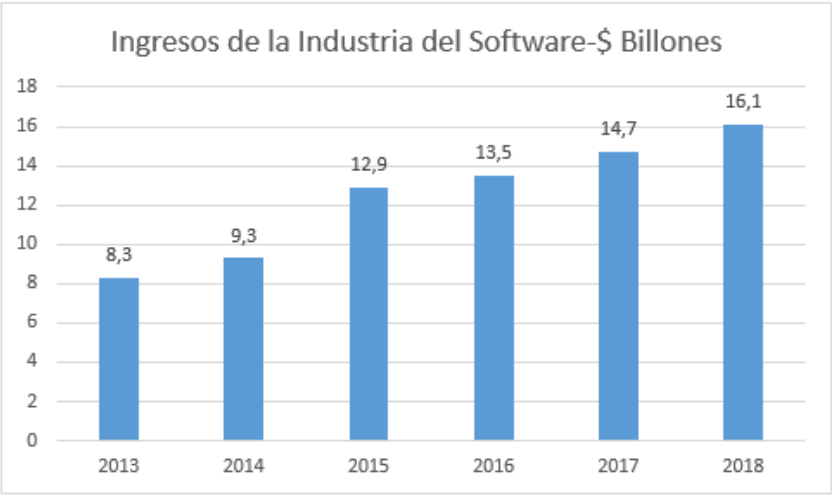


Figura 4. Ingresos históricos de la industria del software

Fuentes de Consulta: (Alfonso, 2017; Alfonso, 2018; Dinero, 2015; Portafolio, 2015; Portafolio, 2018)

Alfonso (2017) es clara en afirmar que las empresas del sector de TI no solo aportan significativos dividendos a la economía desde sus ingresos, también generan alrededor de 109.000 empleos en la industria software y otros servicios relacionados, siendo los sectores de salud, financiero y seguridad informática los más beneficiados con el desarrollo de software.

Asimismo, la proyección de las empresas desarrolladoras de software de vender sus productos en el extranjero es bastante significativa. En el 2009 se alcanzó una cifra histórica de repunte al obtener ingresos por USD 35 Millones. Sin embargo al cierre de 2016 las exportaciones de software llegaban a USD 244 Millones y se espera que en el 2018 se alcance una cifra cercana a los USD 260 Millones.

Año	Exportaciones (USD Millones)
2013	238
2014	241
2015	237
2016	244
2017	248
2018	260

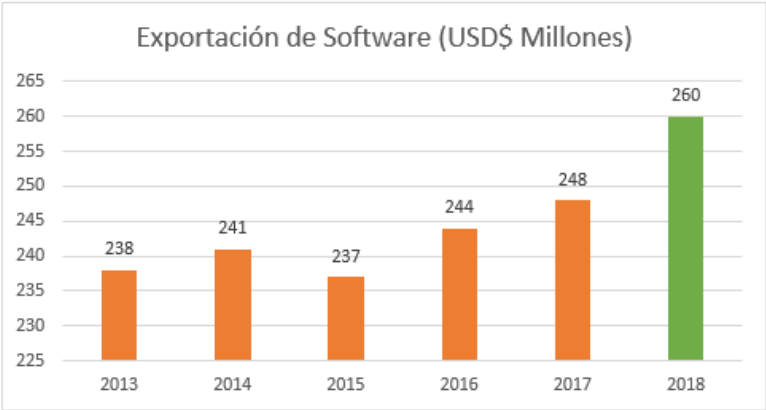


Figura 5. Exportaciones históricas de software (USD Millones)

Fuentes de consulta: (Fedesoft, 2015; Portafolio, 2018; UNP, 2015)

El estudio de Portafolio (2015) deja claro como en el 2015 hubo una disminución de los ingresos por exportación de software. Las empresas grandes del negocio de software vieron afectada su utilidad neta producto de la devaluación del peso colombiano, que encareció sus obligaciones financieras en el extranjero.

Los países receptivos del software hecho en Colombia son Estados Unidos, Ecuador y España; mercado que se mantiene y sigue consolidándose como el principal destino de los aplicativos informáticos hechos en Colombia.

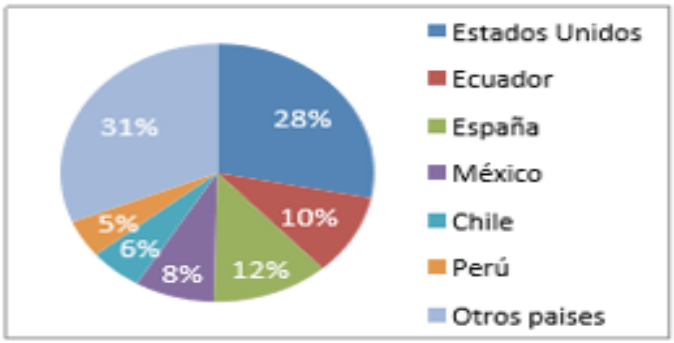


Figura 6. Países donde se exporta el producto software hecho en Colombia

Fuente: (Fedesoft, 2015)

5.2. Metodologías de desarrollo de software

A medida que el software se fue posicionando como activos intangibles en las empresas, surgieron metodologías que formalizaban su elaboración y establecían lineamientos para su desarrollo. Se destacan metodologías tradicionales y metodologías ágiles, cada una con características específicas que ofrecen diversos modelos estratégicos para los desarrollos.

5.2.1. Metodologías tradicionales

Las metodologías tradicionales son aquellas que se enfocan en la planeación y control a largo plazo de los proyectos. Por ser rígidas también se les llaman metodologías pesadas.

Esta apreciación se debe a que son metodologías basadas en una planificación total, de principio a fin del proyecto, con controles tan bien definidos como los roles y actividades. Esta estricta planificación no ofrece facilidad a los cambios, por lo que una característica innata de las metodologías tradicionales es la resistencia a ellos.
Dentro de las metodologías tradicionales más usadas están:

- Metodología en V (Verificación y Validación)
- Metodología en cascada
- Modelo de ciclo de vida en espiral
- Modelo Rational Unified Process (RUP)

5.2.1.1. Metodología en V (Verificación y Validación)

Grados y La Serna (2015) enuncian que el proceso de Verificación y Validación (V&V) del software, se inicia con el levantamiento de requisitos de un proyecto y culmina con el paso a producción del aplicativo, sin ser un proceso aislado que solo se active con la culminación del desarrollo del software, sino que debe estar presente en todas las fases del desarrollo e integrado con las necesidades del negocio

Anteriormente Britto y Dapozo (2011) habían aseverado que las actividades de V&V tienen como objetivo asegurar que el software satisface los requisitos del usuario y la calidad esperada donde La Verificación debe responder a la pregunta ¿Estamos construyendo el producto correctamente?, y la Validación a la pregunta ¿Estamos construyendo el producto correcto?

Desde la perspectiva de Zamora (2011) la Verificación debe determinar si el producto resultante de una fase del ciclo de vida software cumple los requisitos establecidos en la fase anterior y además es completo, consistente y correcto para comenzar la siguiente fase. El proceso de Validación complementa indicando si el software cumple su especificación comportándose como se espera conforme las expectativas del cliente.

Dentro del proceso de Verificación y Validación se utilizan dos técnicas de comprobación y análisis de sistemas, Drake y López (2009) afirmaron:

- Las inspecciones del software analizan y comprueban las representaciones del sistema como el documento de requerimientos, los diagramas de diseño y el

código fuente del programa y son aplicadas rigurosamente a. todas las etapas del proceso de desarrollo.

- Como complemento, las pruebas del software consiste en comprobar el comportamiento del programa a series de datos de prueba y examinar sus respuestas y comportamiento operacional. Ejecutar las pruebas es una técnica dinámica de V&V e impone la necesidad de disponer de un prototipo ejecutable del sistema.

5.2.1.2. Metodología en cascada

Es el modelo que se ha seguido por mucho tiempo en el desarrollo de software tradicional hasta la aparición de otras tendencias. Su planteamiento sugiere una secuencia serial entre fases como se muestra en la Figura 5. Su mayor desventaja es que no planteó las retroalimentaciones de errores encontrados en una fase que tuvieran sus causas en el Análisis y Diseño sino en la etapa de programación evidenciadas en la fase de Pruebas o de Mantenimiento.

Las fases de Análisis y Diseño se limitan en el tiempo, sin embargo una vez entregado el producto final y perpetuado el mantenimiento, el software se mantiene en un ciclo constante de correcciones o adaptaciones a los cuales debe efectuarse su correspondiente análisis y diseño.

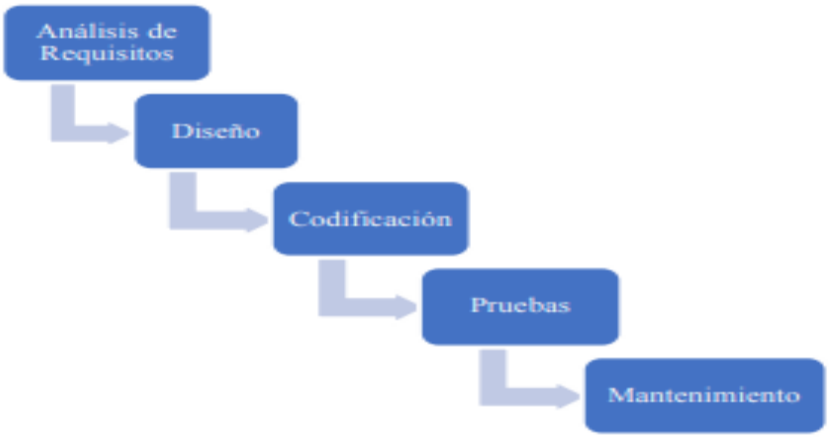


Figura 7. Modelo en cascada de desarrollo de software

Fuente: (Molina, Vite, y Dávila, 2018)

5.2.1.3. Modelo de ciclo de vida en espiral.

Almunia (2004) afirma que Barry Boehm, definió este modelo enfocado en los desarrollos rápidos partiendo de un levantamiento no formal de los requisitos y desarrollando un prototipo funcional del software de acuerdo a un interés o necesidad particular. Cada necesidad adicional incrementa el software ya culminado y se aplica en proyectos con requerimientos no definidos claramente, presentando como principal defecto el que no se ejerce mayor control sobre el desarrollo y por consiguiente la documentación sobre el mismo es muchas veces inexistente o insuficiente por la velocidad en que se desarrolla.

Las fases identificadas en este modelo son:

- Determinación de objetivos, medidas y restricciones
- Evaluar alternativas, identificar y resolver riesgos
- Desarrollar e identificar producto del siguiente nivel

- Planificar siguientes fases.

La falta de requerimientos definidos y el poco control sobre el desarrollo genera una planeación incierta de los costos los cuales se van conociendo a medida del desarrollo al igual que el tamaño exacto del software. Adicionalmente los mantenimientos se convierten en un problema complejo por la falta de documentación.

5.2.1.4. Modelo Rational Unified Process (RUP)

Una variante del Modelo en Espiral es el Rational Unified Process (RUP). Este modelo es comercializado por Rational Software, una compañía propiedad de IBM. Se enfoca principalmente en desarrollo de software orientado a objetos.
Implementa seis de las mejores prácticas en el desarrollo de software actual:

- **Gestión de requisitos**

  Se encarga del proceso de búsqueda, organización y control de cambios y configuración de los requisitos, llamados funcionalmente Casos de uso.

- **Desarrollo de software iterativo.**

  Se definen hitos dentro de todo el desarrollo y se aplica un número de actividades a cada uno haciendo énfasis en la naturaleza de la actividad.

- **Desarrollo basado en componentes**

  Se divide el software en partes o componentes, definiendo adecuadamente sus interfaces y ensamblando posteriormente

- **Uso de Unified Modeling Language (UML)**

  Dado que el modelo de desarrollo impone un desafío al equipo de trabajo debido a su complejidad, el Lenguaje Unificado de modelos de sistemas o UML (por sus siglas en inglés), por ser un lenguaje especializado en la visualización, construcción y documentación de sus componentes, ayuda a la comprensión del sistema desarrollado.

- **Verificación continua de la calidad.**

  Durante el proceso de desarrollo, la calidad de los componentes es evaluada periódicamente, en especial al final de cada iteración.

- **Gestión de cambios y configuración**

  Es una disciplina importante en este modelo de desarrollo. Actualmente, la dispersión geográfica de los desarrolladores es cada vez más común y el registro de los cambios sobre los requerimientos iniciales obliga a gestionar cada cambio y el impacto sobre la configuración del sistema para evitar un caos en la calidad del producto final.

RUP divide el proceso en cuatro Fases que interactúan con nueve disciplinas. Las fases. No se sugiere una asociación específica entre una disciplina y una fase.

5.2.2. Metodologías ágiles

El empleo de las metodologías tradicionales en proyectos de desarrollo de software con poca información de requerimientos a largo plazo presentaban serios problemas debido a que no era posible la planificación a largo plazo y por ende carecían de controles eficaces para el monitoreo de su avance.

En la década de los noventa surgieron metodologías de desarrollo de software ligeras, que de acuerdo a Navarro, Fernández y Morales (2013), buscaban reducir la probabilidad de fracaso en proyectos largos debido a la subestimación de costos, tiempos y funcionalidades. Estas metodologías nacieron como reacción a la burocracia inherente a las metodologías tradicionales en los proyectos de pequeña y mediana escala.

Como lo resumen Canós, Letelier y Panadés (2003), en febrero de 2001 como resultado de una reunión celebrada en Utah (EEUU) con la participación de expertos de la industria, nace el término “ágil” aplicado al desarrollo de software. Su objetivo fue esbozar los valores y principios que deberían permitir a los equipos desarrollar software de manera rápida y que se pudiera responder a los cambios durante la ejecución del proyecto.

De esta reunión surgió el “Manifiesto Ágil”, un documento que establece valores y principios que se deben cumplir al emplear esta filosofía.

Tabla 1. *Valores y principios del manifiesto ágil*

VALORES Y PRINCIPIOS DEL MANIFIESTO ÁGIL	
Valores	• Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas. La gente es el principal factor de éxito de un proyecto software. • Es más importante construir un buen equipo que construir el entorno. Muchas veces se comete el error de construir primero el entorno y esperar que el equipo se adapte automáticamente. Es mejor crear el equipo y que éste configure su propio entorno de desarrollo en base a sus necesidades. • Desarrollar software que funciona más que conseguir una buena documentación. La regla a seguir es "no producir documentos a menos que sean necesarios de forma inmediata para tomar un decisión importante". Estos documentos deben ser cortos y centrarse en lo fundamental. • La colaboración con el cliente más que la negociación de un contrato. Se propone que exista una interacción constante entre el cliente y el equipo de desarrollo. Esta colaboración entre ambos será la que marque la marcha del proyecto y asegure su éxito. • Responder a los cambios más que seguir estrictamente un plan. La habilidad de responder a los cambios que puedan surgir a los largo del proyecto (cambios en los requisitos, en la tecnología, en el equipo, etc.) determina también el éxito o fracaso del mismo. Por lo tanto, la planificación no debe ser estricta sino flexible y abierta.
Principios	• La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor. • Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva. • Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas. • La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto. • Construir el proyecto en torno a individuos motivados. Darles el entorno y el apoyo que necesitan y confiar en ellos para conseguir finalizar el trabajo. • El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo. • El software que funciona es la medida principal de progreso. • Los procesos ágiles promueven un desarrollo sostenible. Los promotores, desarrolladores y usuarios deberían ser capaces de mantener una paz constante.

	• La atención continua a la calidad técnica y al buen diseño mejora la agilidad. • La simplicidad es esencial. • Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos. • En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

Fuente: (Canós *et al*., 2003)

Más adelante y como consecuencia del nacimiento del Manifiesto Ágil, las metodologías ligeras adoptan el nombre de Metodologías Ágiles (MA), las cuales, como lo manifiestan Navarro, Moreno, Aranda, Parra, Rueda, y Pantano (2017), han impuesto su uso para el desarrollo de software. La razón de esta tendencia es la poca eficiencia que han demostrado las metodologías tradicionales para atender los cambios en los requerimientos de clientes y usuarios, limitando la competitividad y a la obtención de mayores beneficios en la producción de bienes o en la prestación de servicios, en el menor tiempo posible.

Según Rodríguez (2015), las ventajas del uso de las metodologías ágiles son:

- Respuesta rápida tanto a cambios de requisitos durante el proyecto como a la detección de errores por la manera en que se atienden las pruebas a medida que se avanza.
- Minimiza costos y mejora ostensiblemente la velocidad y eficiencia del desarrollo del software.
- El equipo de desarrollo conoce el estado del proyecto.
- Mejora la calidad del producto.
- Simplifica la sobrecarga de procesos.

5.2.2.1. Scrum

Como la definen sus creadores, es un marco de trabajo para desarrollar, entregar y mantener productos complejos (Shwaber y Sutherland, 2017)

García (2015) es claro en afirmar que las características más relevantes que se logran en un proyecto basado en Scrum son: gestión regular de las expectativas del cliente, resultados anticipados, flexibilidad y adaptación, retorno de la inversión, mitigación de riesgos, productividad y calidad, alineamiento entre el cliente y el equipo y un equipo motivado.

Scrum presenta tres componentes principales: Equipo de trabajo, eventos y artefactos.

Equipo de Trabajo**.** El equipo de trabajo está conformado por el Dueño del producto, Scrum Master y el Equipo de desarrollo.

Dueño del producto: "Es el responsable de maximizar el valor del producto resultante del trabajo del Equipo de Desarrollo" y único responsable de gestionar la Lista del Producto y es representado por una persona, no por un comité. (Shwaber y Sutherland, 2017). La autonomía es clave para el ejercicio de su labor y sus decisiones se reflejan en la Lista de productos, solamente él puede gestionar sus cambios.

Scrum Master: Es un líder al servicio de todo el equipo Scrum y es responsable de asegurar que se ejecuta Scrum como se define en la Guía ayudando a todos a entender la teoría, prácticas, reglas y valores de Scrum y fortaleciendo las relaciones con todos los que interactúan con la metodología. (Shwaber y Sutherland, 2017). El Scrum Master no solo está al servicio del Dueño del Producto y el Equipo de

Desarrollo, también interactúa con la Organización para la cual se desarrolla el proyecto.

Equipo de Desarrollo: Son los profesionales encargados de realizar el trabajo y entregar un producto terminado que se puede pasar a producción al final de cada Sprint. Es indispensable que solo el Equipo de Trabajo participe en la elaboración del incremento, así lo define la Guía de Scrum.

Shwaber y Sutherland (2017) establecen características para los equipos de trabajo:

- Son auto organizados y nadie les puede indicar cómo convertir elementos de la Lista del Producto en Incremento.
- Son Multifuncionales. Tienen todas las habilidades para elaborar un incremento.
- No se reconocen rangos ni subequipos, todos son responsables del resultado al final del sprint.
- El tamaño del equipo de trabajo no se especifica pero equipos muy pequeños (menos de 3 personas) pueden tener complicaciones en cuanto a habilidades, aunque desarrollan mejor comunicación. Los equipos grandes (más de 9 miembros) requieren demasiada coordinación, aunque es fácil encontrar habilidades diversas.

Eventos Scrum.

Sprint. Es el ciclo de trabajo de máximo 4 semanas el cual debe cumplir un objetivo que no puede ser modificado durante su ejecución y en el cual al final se desarrolla un incremento y .Dentro del sprint se desarrollan los siguientes eventos:

Planeación del Sprint. Según Shwaber y Sutherland (2017), el resultado es el plan que se crea mediante el trabajo colaborativo del Equipo Scrum completo y su objetivo es responder a las preguntas:

- ¿Qué puede entregarse en el Incremento resultante del Sprint que comienza?
- ¿Cómo se conseguirá hacer el trabajo necesario para entregar el Incremento?

Scrum Diario: Como lo establece Shwaber y Sutherland (2017), “El Scrum Diario es una reunión con un bloque de tiempo de 15 minutos para el Equipo de Desarrollo. El Scrum Diario se lleva a cabo cada día del sprint. En él, el Equipo de Desarrollo planea el trabajo para las siguientes 24 horas”.

Revisión del Sprint. Es la reunión de duración máxima de 4 horas donde se lleva a cabo una inspección del producto terminado. . “Se trata de una reunión informal, no una reunión de seguimiento, y la presentación del Incremento tiene como objetivo facilitar la retroalimentación de información y fomentar la colaboración.” (Shwaber y Sutherland, 2017)

Retrospectiva del Sprint. Es una reunión de máximo 4 horas donde se analizan las situaciones ocurridas al equipo de trabajo durante el sprint. Su objetivo es crear un plan de mejoras basadas en las situaciones e interacciones que pudieran haber afectado el trabajo del Equipo de Desarrollo.

Artefactos de Scrum.

*Lista de Producto***.** En la guía oficial Shwaber y Sutherland (2017), la definen como una lista ordenada de todo lo necesario del producto y cuyo responsable es el Dueño del producto. Esa responsabilidad incluye su contenido, disponibilidad y ordenación.

Lista de Pendientes. Son los elementos de la Lista de Productos seleccionados para formar parte del desarrollo dentro del sprint y conseguir su objetivo. Solo el Equipo de Desarrollo puede cambiar la Lista de Pendientes durante el Sprint.

*Incremento***:** Es el resultado del sprint. Al final de cada sprint el Equipo de Desarrollo entrega un "producto final que sea susceptible de ser ofrecido con el mínimo esfuerzo al cliente cuando lo solicite" (Proyectos Agiles, s.f.).

Una representación gráfica de la metodología se muestra en la siguiente figura.

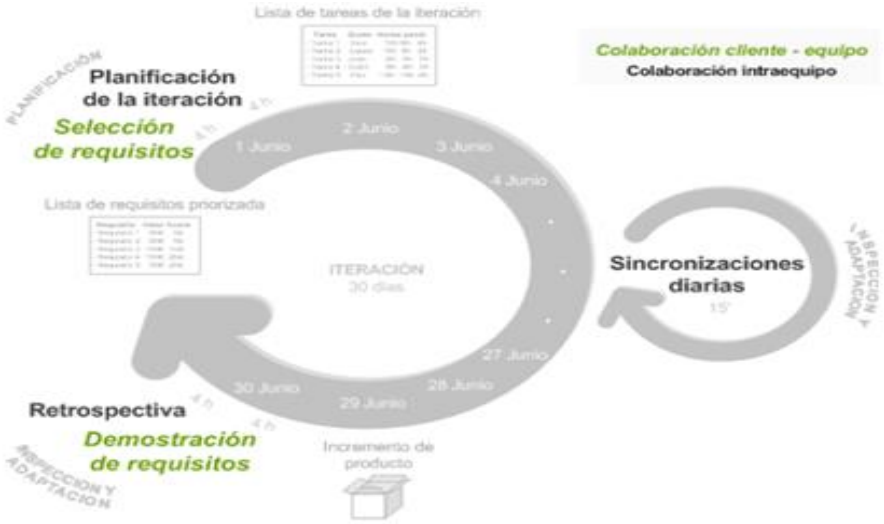


Figura 8. Esquema SCRUM

Fuente: (Proyectos Agiles, s.f.)

5.2.2.2. Extreme Programing (XP)

Como lo expresan Letelier y Panadés (2006), XP es una metodología ágil que se propone fortalecer las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el ambiente colaborativo y la preservación de un buen clima

laboral, basándose en una comunicación en las soluciones y determinación para afrontar los cambios. XP es especialmente sugerida para proyectos con alto riesgo técnico.

El riesgo técnico lo determinan la falta de especificación en los requerimientos, por ello como lo explica Joskowicz (2008) XP propone un ciclo de vida dinámico, donde los clientes no tienen la capacidad de especificar sus requerimientos al comienzo de un proyecto y por ello se establecen ciclos cortos de desarrollo (iteraciones), con entregables funcionales al finalizar cada ciclo. Para cada ciclo se deben llevar a cabo las fases de análisis, diseño, desarrollo y pruebas, pero utilizando un conjunto de reglas propias de XP.

En XP se identifican roles dentro de la ejecución del proyecto. Orjuela y Rojas (2008) resumen los roles principales:

- Cliente: Es el contratante en el proyecto y quien establece los requerimientos de cada desarrollo (Historias de Usuario). Asigna las prioridades de procesos a desarrollar por cada iteración.
- Programadores: se establecen parejas de desarrolladores que rotan en los distintos procesos. Se busca que el conocimiento del proyecto se difunda entre todo el equipo programador.
- Jefe: Relaciona al cliente con los programadores y dentro del equipo de trabajo coordina la ejecución de las actividades.
- Tester: encargado de probar cada parte de aplicativo desarrollado.
- Tracker: realiza el seguimiento a de avance del proyecto en cada iteración y retroalimenta al equipo de trabajo.
- Gerente: es el responsable de todo el proyecto.

Como lo asegura Joskowicz (2008) es una metodología que afianza los vínculos entre sí en todos los interesados en el proyecto (desarrolladores, cliente, gerente) pero presenta su mayor desventaja en la dificultad de estimar el costo del proyecto. Dado que el alcance del mismo no está completamente definido al comienzo, y que la metodología XP es expresamente abierta a los cambios durante todo el proceso, es difícil establecer un presupuesto previo.

5.2.2.3. Kanban

En los años 50 la firma Toyota desarrolló e implementó en su producción automotriz un sistema denominado Manufactura Ágil que sirve para mejorar y optimizar los procesos operativos de cualquier compañía industrial, sin importar su tamaño. Como lo enfatiza Padilla (2010) el objetivo es minimizar el desperdicio. ya sea inventarios, tiempos, productos defectuosos, transporte, almacenajes, maquinaria y hasta personas.

García (2015) expone que la metodología Kanban se basa en el sistema de Manufactura Ágil de Toyota y se establece como objetivo gestionar de manera general como se van completando las tareas de un proyecto de manera visual. Para ello el trabajo se divide en partes, y por lo general cada una de ellas se escribe en tarjetas colocadas sobre un tablero. Las tarjetas suelen tener información variada: descripción, estimación temporal, etc. El objetivo de esta visualización es que quede claro el trabajo a realizar y los estados a medida que se avanza en el proceso

Para cada proceso del proyecto se establecen tantos estados o fases se decida y éstos estados conforman columnas dentro de un tablero y las actividades se pueden organizar en subestados si así se definiera.

Cada columna del tablero Kanban puede discriminar si las tareas de la fase están en estado de Pendiente o Realizada, incluso se pueden establecer más estados si llegara a requerirse.

Un panel Kanban típico se implementaría con tres columnas, que representan las diferentes fases por las que una tarea tiene que fluir para ser desarrollada (análisis, desarrollo y puesta en marcha). Cada fase está subdividida en dos estados, que son “En curso” y “Listo”.

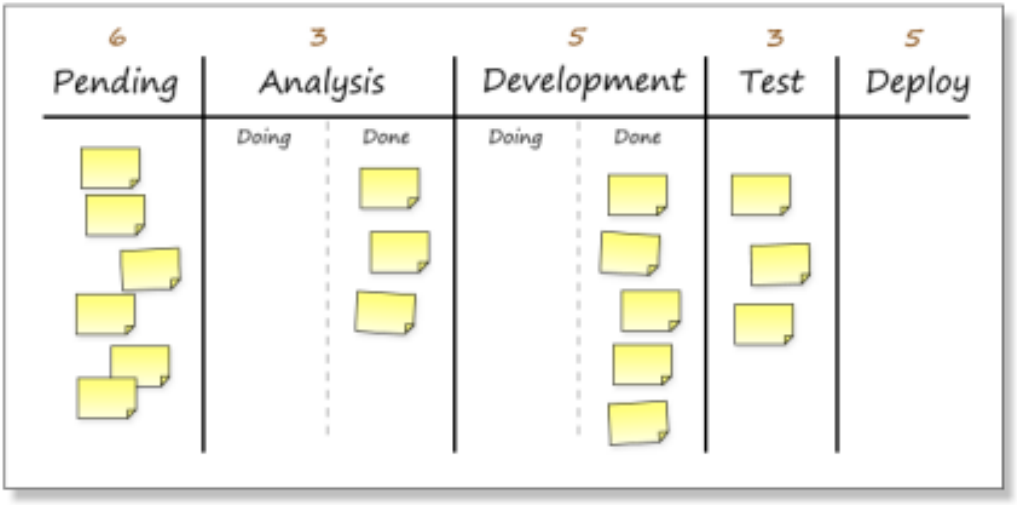


Figura 9. Tablero Kanban

Fuente: (Garcia, 2015)

5.2.2.4. Crystal

Como lo explica Bermejo (2010) es una metodología llamada así haciendo analogía con los minerales, que se caracterizan por dos dimensiones: color y dureza, al igual que los proyecto de desarrollo de software los cuales también pueden caracterizarse

por dos dimensiones: Tamaño (número de personas en el proyecto) y criticidad (consecuencia de los errores).

Las políticas de la Metodología deben adaptarse al equipo de trabajo y sus condiciones. Rodríguez (2008) enfatiza que estas políticas dependerán del tamaño del equipo, estableciéndose una clasificación por colores. Por ejemplo, Crystal Clear, la metodología más ligera de este conjunto, está dirigida a la comunicación de equipos pequeños que desarrollan software cuya criticidad no es elevada mientras que la más compleja es aquella que cuenta con más de 100 miembros participando del proyecto.

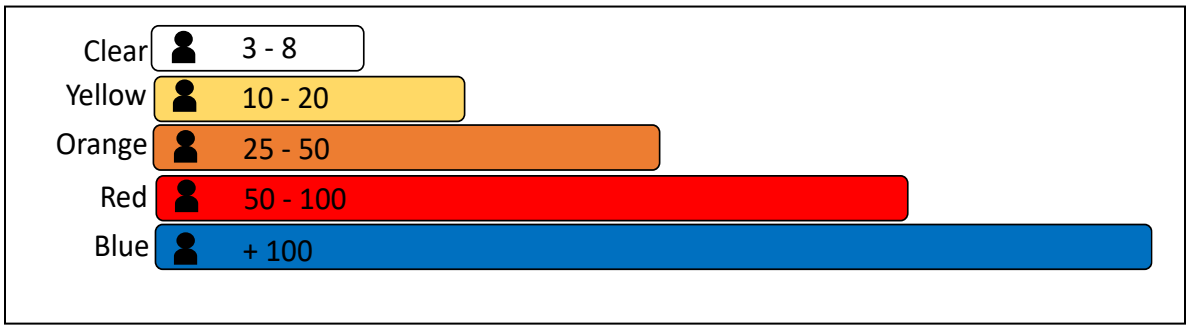


Figura 10. Familia de metodologías Crystal según número de miembros en el proyecto

En general, la familia de metodología Crystal se basa en siete principios y dos reglas. Según García (2015) los principios son:

- Entregas frecuentes: con periodicidad desde semanales hasta trimestrales en modalidad incremental sobre el software en productivo y dentro de un ciclo de trabajo.
- Mejora reflexiva: el trabajo en ciclos o iteraciones ayuda a mejorar el proyecto.
- Comunicación osmótica: la ubicación del equipo de trabajo es la misma para lograr comunicación directa y fluida.

- Seguridad persona: el equipo puede expresar su opinión personal libremente.
- Enfoque: períodos de dos horas de no interrupción al equipo, objetivos y prioridades claros para definir tareas concretas.
- Fácil acceso al cliente: como mínimo una reunión semanal con el cliente y el cliente debe ser accesible.
- Entorno técnico: Pruebas automatizadas, gestión de la configuración e integración continua de los productos resultantes de cada ciclo.

Dos reglas se aplican para toda la familia Crystal. “La primera indica que los ciclos donde se crean los incrementos no deben exceder cuatro meses; la segunda, que es necesario realizar un taller de reflexión después de cada entrega para afinar la metodología” (Navarro *et al*., 2013, p. 36)

5.2.3. Comparativo de las metodologías ágiles con las tradicionales

Como se ha enunciado, la principal diferencia entre las metodologías tradicionales y las ágiles es que las primeras se implementan para proyectos que ofrecen poca tolerancia a los cambios, con una planeación estricta de las actividades y una línea de tiempo extensa. Por el contrario las metodologías ágiles están concebidas para aceptar los cambios y adaptarse de manera rápida a las nuevas condiciones en tiempos de ejecución cortos.

Tabla 2. *Comparación de Metodologías ágiles vs. Metodologías tradicionales*

METODOLOGÍAS ÁGILES	METODOLOGÍAS TRADICIONALES
Son metodologías adaptativas con procesos flexibles y se recibe sin problemas la ocurrencia de cambios durante el proyecto.	Son metodologías predictivas con procesos rígidos que no admiten cambios de gran impacto para el proyecto.
No existe un contrato tradicional que enmarque el desarrollo del software como producto.	Existe un contrato prefijado y el producto debe tener de manera estricta las características contenidas en el documento.
Orientada a Proyectos pequeños, de corta duración.	Aplicables a proyectos de cualquier tamaño pero especializadas en proyectos grandes.
El Proyecto es subdividido en pequeñas partes, cada una con sus entregables en cada etapa.	El proyecto se concibe como un total y al final se ejecuta la entrega del producto.
Existe poca documentación del proceso de desarrollo producto.	Contiene documentación extensa del desarrollo del producto.
Está orientada al aspecto humano: El individuo y el trabajo en equipo	Orientado a la definición de procesos: Roles, actividades y artefactos
Se basa en un conocimiento empírico proveniente de la producción del código	Basadas en normas provenientes de estándares seguidos por el entorno de desarrollo
El cliente es parte del equipo de desarrollo (Cliente in situ) y se establece una comunicación continua y permanente	El cliente interactúa con el equipo de desarrollo mediante reuniones periódicas.

Fuente: (Letelier y Panadés, 2006; Navarro *et al*., 2013;)

5.3. Elicitación de necesidades

Los proyectos de desarrollo de software posterior a la conformación de los equipos de trabajo, inician la fase de levantamiento de información. Se refiere a que el cliente exponga al equipo desarrollador las necesidades que deben ser cubiertas por él.

La salida de este proceso es la identificación de las necesidades funcionales y las no funcionales del software a desarrollar.

Las necesidades funcionales son las descripciones detalladas de los procesos que debe proveer el software, más concretamente son los detalles del comportamiento del sistema ante entradas y la información que manejará y entregará como respuesta. Estas necesidades funcionales ingresarán al proceso de desarrollo de software directamente y es responsabilidad del desarrollador hacerlas cumplir dentro del código del aplicativo.

Por otro lado las necesidades no funcionales son las que no se refieren explícitamente a lo que el sistema debe hacer sino a condiciones impuestas por situaciones externas determinadas por el usuario o la organización a la que se le desarrolla el software, por ejemplo limitaciones de presupuesto, tecnología para ejecutar procesos específicos y orientan su implementación hacia las cualidades (también llamadas atributos) que el software debe tener.

En el desarrollo de software a las necesidades no funcionales se les conoce como atributos de calidad. Como lo enuncia Alfaro (s.f.) existen atributos de calidad que se deben cuidar en el desarrollo de todo producto de software y se puede agrupar de acuerdo al interés del usuario o del desarrollador, sin embargo no existe una regla que deba cumplirse en este sentido.

Algunos de los atributos de calidad empleados en el desarrollo de software son: Eficiencia, Seguridad, Flexibilidad, Confiablidad, Disponibilidad, Mantenibilidad, Usabilidad, Integridad, Gestionabilidad y Confidencialidad

Para el manejo de los atributos de calidad se emplean los Árboles de Utilidad y los Escenarios de calidad.

5.4. La seguridad informática

La protección de la información es un tema que tiene muchas aristas donde cada una enfoca sus esfuerzos en establecer controles que permitan resguardar la integridad y confiabilidad de los datos.

Para reducir el enfoque en este trabajo, se hablará de la seguridad de la información digital, aquella que es accedida y manejada de manera exclusiva por aplicaciones de computador.

Una debilidad en un aplicativo informático es la situación riesgosa no tenida en cuenta dentro de los controles establecidos para fortalecer la seguridad de la información. De esta debilidad surge la vulnerabilidad o amenaza, definida como la situación potencial que puede afectar o dañar la información manejada a través del aplicativo. Cuando un agente externo se vale de la vulnerabilidad y accede a la información con la intención de daño se produce un ataque.

Como lo afirma Tarazona (2007), “básicamente, podemos agrupar las amenazas a la información en cuatro grandes categorías: Factores humanos (accidentales, errores); fallas en los sistemas de procesamiento de información; desastres naturales y; actos maliciosos o malintencionados”.

Es responsabilidad de los Departamentos de Informática de las empresas proveer controles y establecer mecanismos para salvaguardar la integridad de la información ante desastres naturales o fallas de los elementos tangibles (Hardware) donde se almacena la información o ejecutan sus procesos.

Sin embargo, las fallas en los sistemas de procesamiento, la amenaza de los actos maliciosos y los errores accidentales en el manejo de los aplicativos son situaciones que deben ser prioridad para las empresas desarrolladoras de software, directamente responsables de entregar al cliente un producto de calidad: funcional, pertinente y seguro.

La funcionalidad y pertinencia se refieren a la correcta ejecución de los procesos teniendo en cuenta los requerimientos del cliente consignados en un documento formal de levantamiento de información. La seguridad, así no haya sido un requisito del cliente, debe proveerla el desarrollador como condición de su ética, experiencia y compromiso en entregar un producto que minimice las oportunidades de ataques y preserve las condiciones de ejecución para mantener la integridad tanto de la información como su código fuente.

En este trabajo se abordarán únicamente las amenazas relacionadas a los aplicativos desarrollados bajo la arquitectura web.

Los ataques a la información se pueden ejecutar de manera directa a las bases de datos o a través del código fuente de los aplicativos que la manejan. Se han identificado patrones de ataques y establecidos controles para reducir el efecto de los mismos.

De acuerdo con Matei (2015) uno de los métodos clave que permite mejorar la estabilidad, el rendimiento y la seguridad de una arquitectura de software es el aprovechamiento de los patrones probados, de ahí que el acierto en la selección de los patrones de diseño y patrones de seguridad ofrece una significativa mitigación de los riesgos.

5.5. La seguridad en aplicaciones web

Un aspecto importante que deben tener en cuenta las empresas desarrolladoras de software de arquitectura web es que “la tecnología evoluciona de igual manera como lo hacen sus amenazas, y puede verse vulnerable en el día a día” (Guerrero *et al*., 2017, p. 216)

Una práctica que permite el crecimiento de los riesgos es la falta de mantenimiento de los aplicativos que son liberados y que forman parte del sistema en productivo. Guerrero et al. (2017) es claro en afirmar que en la mayoría de los casos la reducción en los tiempos de lanzamiento de las aplicaciones web y el enfoque a una correcta funcionalidad dejando de lado la seguridad puede generar un producto vulnerable.

Esas debilidades del software son aprovechadas desde Internet para ejecutar acciones dañinas en contra de la información. Los niveles de peligrosidad de los incidentes se muestran a continuación.

CRITERIOS DEL NIVEL DE PELIGROSIDAD DE LOS CIBERINCIDENTES			
NIVEL	**AMENAZAS SUB-YACENTES MÁS HABITUALES**	**VECTOR DE ATAQUE**	**CARACTERÍSTICAS POTENCIALES DEL CIBERINCIDENTE**
CRÍTICO	Ciberespionaje	*APTs*, campañas de *malware*, interrupción de servicios, compromiso de sistemas de control industrial, incidentes especiales, etcétera.	- Capacidad para exfiltrar información muy valiosa, en cantidad y en poco tiempo. - Capacidad para tomar el control de los sistemas sensibles, en cantidad y en poco tiempo.
MUY ALTO	- Interrupción de los - Servicios IT. Exfiltración de datos. - Compromiso de los Servicios.	- Códigos dañinos confirmados de Alto Impacto (RAT, troyanos enviando datos, *rootkit*, etcétera). - Ataques externos con éxito.	- Capacidad para exfiltrar información valiosa, en cantidad apreciable. - Capacidad para tomar el control de los sistemas sensibles considerable.
ALTO	- Toma de control de los sistemas. - Robo y publicación o venta de información sustraída. - Ciberdelito. - Suplantación.	- Códigos dañinos medio impacto (virus, gusanos, troyanos). - Ataques externos compromiso de servicios no esenciales (*DoS/DDoS*). - Tráfico DNS con dominios APTs o campañas *malware*. Accesos no autorizados. - Suplantación. - Sabotaje. - *Cross-Site Scripting*. - Inyección *SQL*. - Spear *phising/pharming*.	- Capacidad para exfiltrar información valiosa. - Capacidad para tomar el control de ciertos sistemas.
MEDIO	- Logro o incremento de capacidades ofensivas. - Desfiguración de páginas web. - Manipulación de información.	- Descargas de archivos sospechosos - Contactos con dominios o direcciones IP sospechosas - Escáneres de vulnerabilidades. - Códigos dañinos de bajo impacto (*adware*, *spyware*, etcétera). - *Sniffing*. - Ingeniería social.	- Capacidad para exfiltrar un volumen apreciable de información. - Capacidad para tomar el control de algún sistema.
BAJO	- Ataques a la imagen. - Errores y fallos.	- Políticas. - *Spam* sin adjuntos. - *Software* desactualizado. - Acoso, coacción, comentarios ofensivos. - Error humano. - Fallo HW-SW.	- Escasa capacidad para exfiltrar volumen de información. - Nula o escasa capacidad para tomar el control de sistemas.

Figura 11. Nivel de peligrosidad de los ataques provenientes de Internet

Fuente: (Villalba y Corchado, 2017)

En todo caso, para cualquier lenguaje que se emplee en la elaboración del software, López (2015) establece que se deben establecer controles que incrementen el nivel de seguridad del producto sobre los siguientes incidentes:

- Desbordamiento de Buffer. Ocurre cuando un programa escribe datos en memoria más allá de la memoria que tenía reservada, como consecuencia de esto se puede sobrescribir el código que se está ejecutando en la máquina.
- Validación de datos. Comprobar la validez de los datos que son entregados a una aplicación evita que esta pueda llegar a un estado inesperado y actuar de forma imprevista, cobrando mayor importancia hacerla en las aplicaciones que actúan como servidor.
- Asignación y acceso a memoria. Una mala asignación de memoria puede llevar a que se accedan a datos que en otras condiciones no podrían ser accedidos, o incluso enviados al exterior. Asignar memoria erróneamente puede ocasionar consultas a todo el contenido de la memoria RAM de un equipo.
- Inyección de código. Consiste en enviar como parámetro una cadena de texto que representa código con el fin que la aplicación que recibe dicho parámetro lo ejecute. Los casos más comunes de este tipo de fallo se dan en forma de Cross Site Scripting (también conocido como XSS).
- Formato de los datos. Se deben llevar a las mismas unidades de medidas los datos que maneja la aplicación. Hacer esto además de mejorar la seguridad del proyecto también nos permite ahorrar costos al no tener que programar, ni utilizar, rutinas que hagan una traducción entre los diferentes formatos.

- Criptografía. Si se van a emplear algoritmos criptográficos, éstos deben ser fuertes. Utilizar algoritmos débiles nos puede llevar a una falsa sensación de seguridad y es posible que terceros sean capaces de obtener información que pensábamos se estaba transmitiendo de forma segura.
- Herramientas de análisis de código estático. El uso de herramientas de análisis de código estático ayuda a localizar puntos vulnerables del código durante la implementación del proyecto.
- Creación de logs. La creación y almacenamiento de logs permite rastrear el origen del problema y corregirlo ante la ocurrencia de un fallo.

Uno de los métodos que han impreso efectividad para evitar los accesos no autorizados es el denominado *Completely Automated Public Turing test to tell Computers and Humans Apart* (Prueba de Turing completamente automática y pública para diferenciar ordenadores de humanos) cuya sigla es Captcha. Muñoz y García (2015) lo definen como un test controlado por una máquina y consiste en que el usuario introduzca correctamente un conjunto de caracteres que se muestran en una imagen distorsionada que aparece en pantalla.

También se utiliza la selección de imágenes que cumplen con cierto criterio dentro un conjunto de opciones o simplemente aceptar que no se es robot mediante el clic sobre un campo de chequeo.

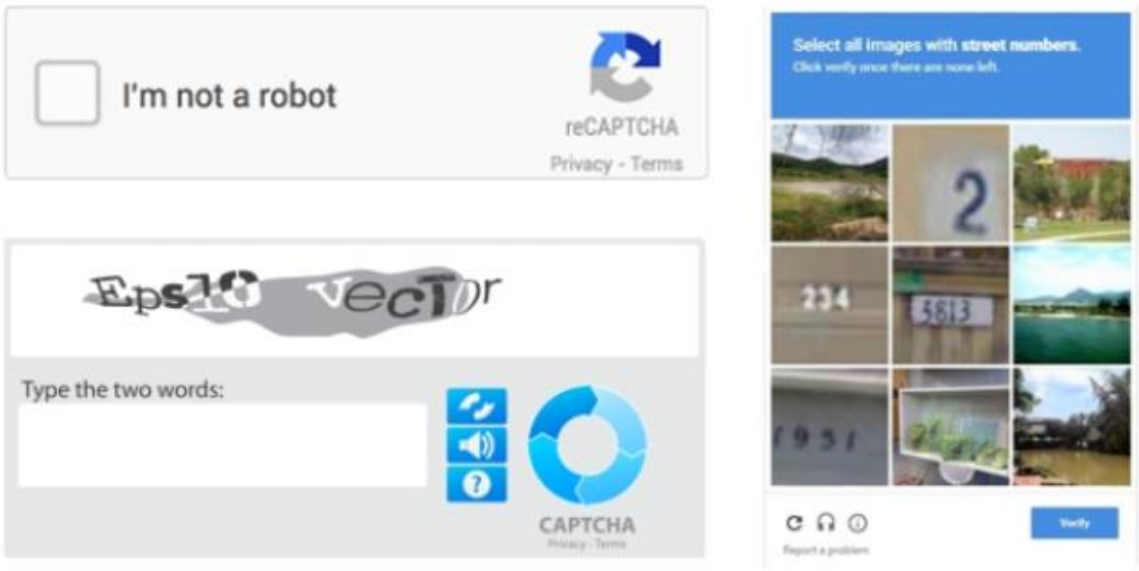


Figura 12. Ejemplos de implementación de Captcha

Fuente: (Excle, 2018)

Su implementación obedece a la masificación del uso de robots en los esquemas de acceso donde se introduce únicamente las credenciales de usuario y contraseña. Con la herramienta Captcha se impone una actividad de raciocinio que solo un humano puede ejecutar. Actualmente muchas variaciones se han implementado en busca de una mayor seguridad en el acceso, incluyendo la inclusión de biometría, lo que sugiere un nivel muy superior al Captcha convencional que conocemos.

5.6. Rol de las pruebas en las metodologías de desarrollo de software

Las pruebas sobre el producto software se deben implementar en todas las fases de su desarrollo. La metodología en “V” para pruebas de software sugiere partir las evaluaciones en dos partes, antes y después de la codificación:

- Evaluaciones estáticas. Son revisiones en las fases previas a la codificación. Cada revisión demarca el inicio de la siguiente actividad.
- Evaluaciones dinámicas. Se denominan pruebas y se realizan sobre el código del programa.

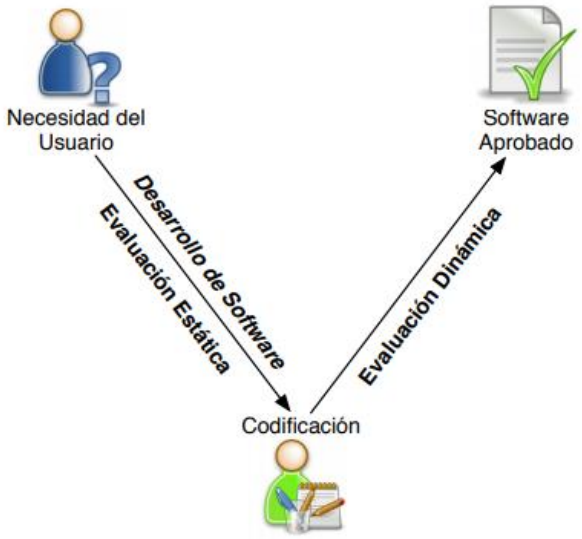


Figura 13. Evaluaciones estáticas y dinámicas en el proceso de desarrollo de software

Fuente: (Fonseca, 2016)

Cuando este modelo de evaluación es detallado, observamos los resultados de cada fase de evaluación y la precedencia de las actividades con respecto al avance del desarrollo del software.

Estas revisiones y pruebas sobre el producto software están encaminadas en conseguir un producto de calidad, no solo constatar la ejecución de manera correcta conforme los requerimientos del cliente sino que también pueda ser eficiente en cuanto al rendimiento y seguridad ofrecida para la preservación de la integridad de la información.

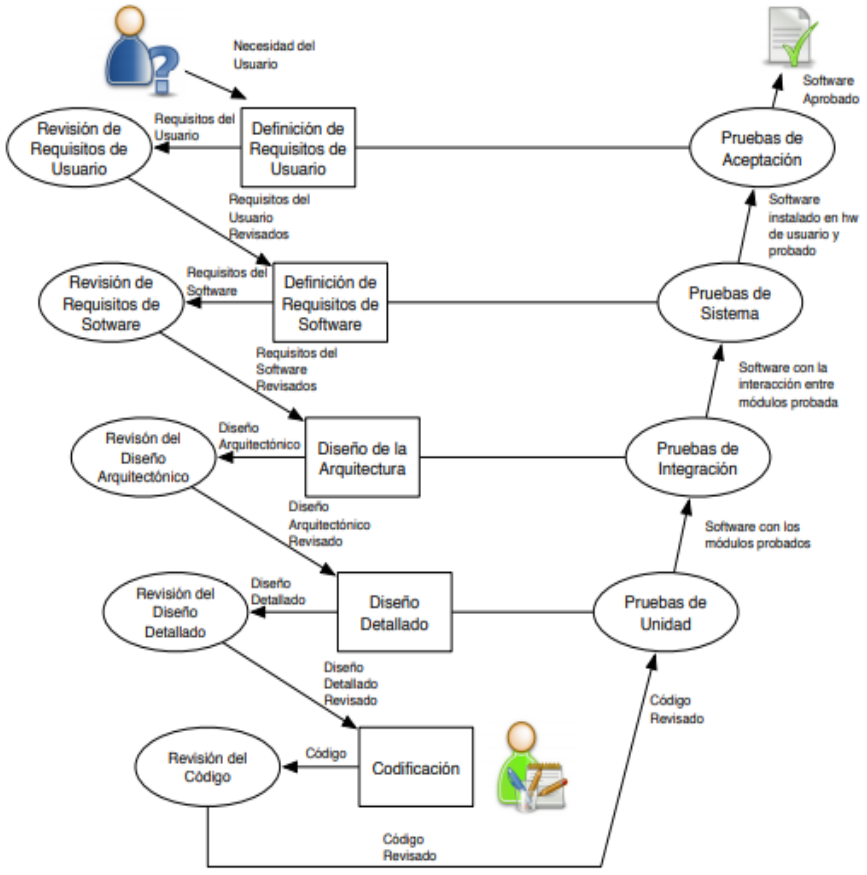


Figura 14. Modelo en V de la evaluación de software

Fuente: (Fonseca, 2016)

Una vez es revisada la codificación y se implementa el programa, se inician los ciclos de pruebas:

- Pruebas de Unidad. Este tipo de prueba se realiza cuando los programas son muy extensos, entonces se los descompone en módulos y se prueba cada una de estas partes. Para este tipo de pruebas se realizan distintos casos de prueba para cada una de las funciones/procedimientos (o métodos). (Díaz *et al.,* 2009)

- Pruebas de Integración. Las pruebas de integración son llevadas a cabo cuando dos o más unidades testeadas son combinadas en una estructura más grande. Esta prueba se realiza en las interfaces entre los componentes y en la estructura más grande que está siendo construida, si su propiedad de calidad no puede ser evaluada a partir de sus componentes. (Fonseca, 2016, p. 40)
- Pruebas del Sistema. Consiste en validar el sistema de principio a fin, basadas en los requisitos funcionales iniciales, aunque los atributos de calidad no funcionales como seguridad y mantenibilidad, son también revisados. (Fonseca, 2016, p. 40)
- Pruebas de Aceptación. Se verifica que el software cumpla con las reglas de negocio. Se describe un escenario de uso del sistema desde la perspectiva del usuario, teniendo en cuenta requisitos funcionales o no funcionales. (Díaz *et al.,* 2009)

5.7. Open Web Application Security Project (OWASP)

Con este panorama, la seguridad en las aplicaciones web adquiere una importancia vital en el desarrollo de los aplicativos. Los riesgos que atentan contra la seguridad se multiplican cuando el aplicativo se encuentra diseñado para acceso web y más aún si está publicado en Internet. Guerrero *et al*. (2017) expresan que para el análisis de estos riesgos y prevenir futuras fugas y fallas en los sistemas software y páginas web es creado OWASP, un foro llamado Open Web Application Security Project (En español Proyecto Abierto de seguridad en aplicaciones Web).

Establecido por OWASP (2018) Uno de los principales proyectos de OWASP es el OWASP Top 10, que se enfoca en identificar los riesgos más críticos para un amplio tipo de organizaciones. Para cada uno de estos riesgos, se proporciona información genérica sobre la probabilidad y el impacto técnico, basado en la metodología de evaluación de riesgos de OWASP.

La metodología propuesta en este trabajo final sugiere que se tengan en cuenta las amenazas y controles contemplados en el OWASP Top 10 vigente. Para el caso actual, se tomarán los documentos OWASP Top 10 2017 y OWASP Top 10 Controles 2018, teniendo que ser actualizada cada vez que una versión más reciente de ellos sea oficializada.

Tal como lo establece OWASP (2018), el documento OWASP Top 10 Controles Proactivos 2018 es una lista de técnicas de seguridad que deberían ser incluidas en todo proyecto de desarrollo de software web y se presentan en orden de importancia siendo el número 1 el más importante.

Tabla 3. *Descripción de Riesgos OWASP Top 10-2017*

RIESGOS	DESCRIPCIÓN
A1: Inyección	Ocurren cuando se envían datos no confiables a un intérprete como parte de un comando o consulta. Los datos dañinos del atacante pueden engañar al intérprete para que ejecute comandos involuntarios o acceda a los datos sin la debida autorización. Ocurren mayormente en SQL, OS, NoSQL, LDAP.
A2: Pérdida de Autenticación	Las funciones de la aplicación relacionadas a autenticación y gestión de sesiones son implementadas incorrectamente, permitiendo a los atacantes comprometer usuarios y

	contraseñas, token de sesiones, o explotar otras fallas de implementación para asumir la identidad de otros usuarios (temporal o permanentemente).
A3: Exposición de datos sensibles	Corresponde a la falta de controles dentro del manejo de los datos sensibles de las aplicaciones, comprometiendo información financiera, de salud o Información Personalmente Identificable (PII). Los datos sensibles requieren métodos de protección adicionales, como el cifrado en almacenamiento y tránsito.
A4: Entidades Externas XML (XXE)	Muchos procesadores XML antiguos o mal configurados evalúan referencias a entidades externas en documentos XML. Las entidades externas pueden utilizarse para revelar archivos internos mediante la URI o archivos internos en servidores no actualizados, escanear puertos de la LAN, ejecutar código de forma remota y realizar ataques de denegación de servicio (DoS).
A5: Pérdida de Control de Acceso	La falta de restricciones a los usuarios correctamente autenticados en el sistema pueden causar ataques debido a la explotación de los defectos para acceder de manera descontrolada a información no permitida, funcionalidades, cuentas de otros usuarios, etc.
A6: Configuración de Seguridad Incorrecta	La configuración de seguridad incorrecta es un problema muy común y se debe en parte a establecer la configuración de forma manual, ad hoc o por omisión (o directamente por la falta de configuración). Son ejemplos: S3 buckets abiertos, cabeceras HTTP mal configuradas, mensajes de error con contenido sensible, falta de parches y actualizaciones, frameworks, dependencias y componentes desactualizados, etc.

A7: Secuencia de comandos de sitios cruzados (XSS)	Los XSS ocurren cuando una aplicación toma datos no confiables y los envía al navegador web sin una validación y codificación apropiada; o actualiza una página web existente con datos suministrados por el usuario utilizando una API que ejecuta JavaScript en el navegador. Permiten ejecutar comandos en el navegador de la víctima y el atacante puede secuestrar una sesión, modificar (defacement) los sitios web, o redireccionar al usuario hacia un sitio malicioso.
A8: Deserialización Insegura	Estos defectos ocurren cuando una aplicación recibe objetos serializados dañinos y estos objetos pueden ser manipulados o borrados por el atacante para realizar ataques de repetición, inyecciones o elevar sus privilegios de ejecución. En el peor de los casos, la deserialización insegura puede conducir a la ejecución remota de código en el servidor.
A9: Componentes con vulnerabilidades conocidas	El uso de bibliotecas, API's o Framework con vulnerabilidades conocidas puede provocar una pérdida de datos o tomar el control del servidor, además de debilitar las defensas de las aplicaciones y permitir diversos ataques e impactos.
A10: Registro y Monitoreo Insuficientes	El registro y monitoreo insuficiente, junto a la falta de respuesta ante incidentes permiten a los atacantes mantener el ataque en el tiempo, pivotear a otros sistemas y manipular, extraer o destruir datos. Los estudios muestran que el tiempo de detección de una brecha de seguridad es mayor a 200 días, siendo típicamente detectado por terceros en lugar de por procesos internos

Fuente: (OWASP, 2018)

Tabla 4. *Descripción de Controles Proactivos OWASP 2018*

CONTROLES	**DESCRIPCIÓN**
C1: Definición de Requerimientos de Seguridad	Los requisitos de seguridad definen nuevas características o adiciones a las características existentes para resolver un problema de seguridad específico o eliminar una vulnerabilidad potencial. Los requisitos permiten a los desarrolladores reutilizar la definición de controles de seguridad y mejores prácticas. El OWASP ASVS (Aplication Security Verification Standard) es un catálogo de requisitos de seguridad y criterios de verificación disponible que se constituye en una fuente de requisitos de seguridad detallada para los equipos de desarrollo.
C2: Aprovechar las librerías y Frameworks seguros	Librerías y Frameworks con codificación segura y seguridad embebida ayudan a los desarrolladores de software a contrarrestar las fallas en los diseños e implementación. Un desarrollador que inicia una aplicación desde cero podría no tener suficiente conocimiento, tiempo o presupuesto para implementar o mantener adecuadamente las características de seguridad. Aprovechar los desarrollos seguros ayuda a lograr los objetivos de seguridad de manera más eficiente y precisa.
C3: Acceso seguro a la base de datos	Se refiere a tener un acceso seguro a todas las bases de datos, tanto Relacionales como NoSQL. Se debe considerar: • Consultas Seguras: Eliminar los riesgos por Inyección. • Configuración Segura: Configurar la Base de datos de manera que guarde seguridad. Seguir una guía de mejores prácticas resulta útil. • Autenticación Segura: Todos los accesos a la Base de Datos deben ser autenticados sobre un canal seguro.

	• Comunicación Segura: La comunicación con el Data Base Management System (DBMS) debe estar encriptada. •
C4: Codificación y Escape de Datos	La codificación y el escape son técnicas defensivas destinadas a detener los ataques de inyección. La codificación implica la traducción de caracteres especiales en otros diferentes pero equivalente y que no represente peligro para el intérprete de destino. Este control (o defensa) debe aplicarse justo antes de pasar el contenido al intérprete de destino. El escape de datos implica agregar un carácter especial antes del carácter / cadena para evitar que sea malinterpretado
C5: Validar todas las entradas	La validación de entrada es una técnica de programación que garantiza que solo los datos formateados correctamente pueden ingresar a un componente del sistema o unidad de software. Una aplicación debe verificar que los datos sean sintáctica y semánticamente válidos (en ese orden) antes de usarlo de cualquier manera (incluida la visualización de nuevo al usuario). El empleo de técnicas de Blacklist o Whitelist de la sintáxis ingresada es una buena práctica de seguridad.
C6: Implementar Identidad Digital	La identidad digital es la representación única de un usuario o entidad cuando se involucra en una transacción en línea. La norma NIST 800-63b describe la llamada Authentication Assurance Level (AAL) para tres niveles de autenticación dependiendo de la criticidad de los daros a acceder: Nivel 1 : Passwords Nivel 2: Autenticación Multifactor. Combinación de Password con esquemas biométricos o con Tokens. Nivel 3: Autenticación basada en criptografía. Es requerido cuando el impacto de los sistemas comprometidos podrían

	causar daños personales, pérdidas financieras significativas, daño al público o involucra violaciones civiles o criminales.
C7: Hacer cumplir los controles de acceso	Un control de acceso es el proceso de otorgar o denegar solicitudes en conjunto con asignar o revocar privilegios. Acertar en el diseño de los controles de acceso es vital para la seguridad del sistema.
C8: Proteger los datos en todo momento	Datos sensibles como contraseñas, números de tarjetas de crédito, registros de salud, información personal y los secretos comerciales requieren protección adicional, particularmente si esos datos están bajo las leyes de privacidad u otras regulaciones. Las buenas prácticas implican: • Clasificación de los datos: Establecer niveles de criticidad de acuerdo a la sensibilidad de la información que guardan. • Encriptación de los datos en tránsito: Cuando se entreguen a módulos dentro y fuera del sistema éstos deben viajar encriptados. • Encriptación al almacenar: Dependiendo de la clasificación establecida, algunos datos deben permanecer encriptados en la Base de Datos.
C9: Implementar el registro y monitoreo de seguridad	Son dos procesos que fortalecen la seguridad en la ejecución. El registro se refiere a grabar información durante el tiempo de ejecución del aplicativo en puntos clave o críticos. El monitoreo es la revisión en vivo de la aplicación y los registros de seguridad usando varias maneras de automatización.
C10: Manejar todos los errores y excepciones	El manejo de excepciones es un concepto de programación que permite que una aplicación responda a diferentes estados

	de error (como la red inactiva, o la conexión a la base de datos falló, etc.) de manera correcta sin afectar su ejecución. Manejar las excepciones y los errores correctamente es fundamental para hacer que su código sea confiable y seguro. El manejo de errores también es importante desde una perspectiva de detección de intrusos. Ciertos ataques pueden desencadenar errores que pueden ayudar a detectar ataques en progreso.

Fuente: (OWASP, 2018)

5.8. Ciclos de vida de desarrollos de software seguro

El SDLC (Security Development Lifecycle) es una metodología desarrollada por Microsoft que aplica el modelo en cascada y facilita a los desarrolladores la elaboración de aplicativos más seguros con el cumplimiento de requisitos y (Microsoft, s.f.). Una de las razones que se expone para esta metodología es que los costos también se ven reducidos si se tienen en cuenta los controles durante el ciclo de desarrollo. El costo de solucionar vulnerabilidades en el software es mucho mayor en la etapa productiva que en una etapa anterior, de hecho es mucho menor el costo cuando se detectan en las fases de análisis y diseño del software.

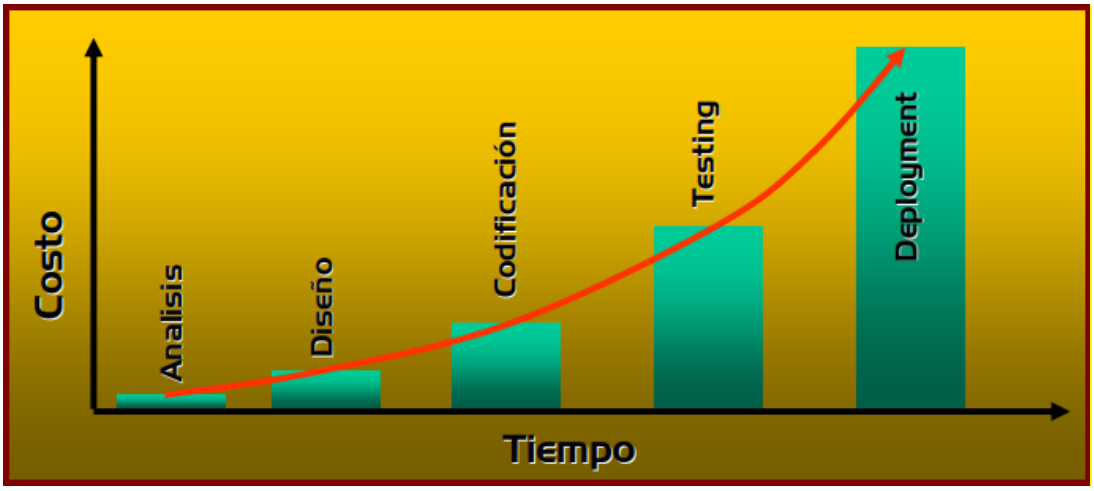


Figura 15. Costo de solución de vulnerabilidades v.s. etapa del software

Fuente: (Milano, 2007)

Para conseguir este objetivo se proponen 17 prácticas agrupadas en 7 fases.

Tabla 5*. Fases y Prácticas de SDLC*

FASE	**PRACTICAS**
Entrenamiento	#1. Entrenamiento básico de seguridad
Requisitos	#2. Establecer Requisitos de Privacidad y Seguridad #3 Crear puertas de calidad #4. Realizar evaluaciones de riegos de seguridad y privacidad
Diseño	#5. Establecer requisitos de diseño #6. Análisis y reducción de superficies de ataques #7. Usar la modelación de amenazas
Implementación	#8. Utilizar las herramientas aprobadas #9. Depreciar funciones inseguras #10. Realizar análisis estático.

Verificación	#11. Realizar análisis dinámico
	#12. Pruebas de Fuzzing
	#13. Revisión de superficie de ataque
Liberación	#14. Crear un plan de respuesta a incidentes
	#15. Realizar la vision final de seguridad
	16. Certificar lanzamiento y archivado
Respuesta	#17. Ejecutar el plan de respuestas a incidentes

Fuente: (Microsoft, s.f.)

6. Caracterización de la metodología propuesta.

Este trabajo tiene como objetivo proponer una metodología que fusione las funcionalidades de una metodología ágil para el desarrollo de software web y que a su vez reduzca los riesgos inherentes a este tipo de arquitectura de software.

Los componentes bases de la metodología propuesta son **SCRUM** como marco de trabajo de prácticas ágiles para el desarrollo de software y las consideraciones de **OWASP** para el fortalecimiento de la seguridad de la información para arquitectura de aplicativos web.

6.1. ¿Por qué SCRUM?

En capítulo anterior fueron enunciadas las diferencias entre las metodologías ágiles y las tradicionales, donde la adaptación al cambio constituye la principal característica

que es ventajosa para las metodologías ágiles. La razón: en el desarrollo de software las modificaciones sobre los requerimientos e información obtenida en el levantamiento inicial son permanentes. Además es mucho más palpable el efecto o impacto del software en la productividad de la empresa si se desarrolla por módulos que se puedan ir apreciando de manera creciente y no esperar intervalos grandes de tiempo para iniciar un proceso en productivo.

Muñoz, Mejía y Corona (2016) afirman que “hoy en día debido a la creciente importancia del software en diversos dominios, existe un gran interés por las Pymes desarrolladoras de software por la adopción de tecnologías ágiles con la finalidad de desarrollar software tan pronto como sea posible”.

Queda claro entonces que la metodología propuesta optará la base de una metodología ágil para ser implementada. Pero, ¿Cuál es la mejor opción?

Más allá de las preferencias a la hora de escoger una metodología, la decisión se basa en la tendencia del mercado reflejado en los estudios sobre lo que las empresas desarrolladoras han implementado.

VersionOne es una herramienta de gestión ágil “todo en uno” construida desde cero para soportar metodologías ágiles de desarrollo de software como Scrum, Kanban, Lean, XP e Hybrid. La organización ayuda a las empresas a visualizar y ofrecer un excelente software. Desde su creación en 2002, VersionOne se ha centrado en gran medida en la promoción y el servicio de la comunidad de desarrollo de software ágil.

Anualmente VersionOne.com lleva a cabo un estudio sobre las metodologías ágiles usadas por la industria del software dentro de su herramienta líder, de esta manera se emite un informe llamado State of Agile Report que ofrece un resumen estadístico de

gran importancia sobre el empleo y preferencias de las metodologías ágiles en las empresas desarrolladoras de software.

Tomando los tres últimos años se observa que Scrum obtiene una amplia preferencia sobre otras metodologías ágiles.

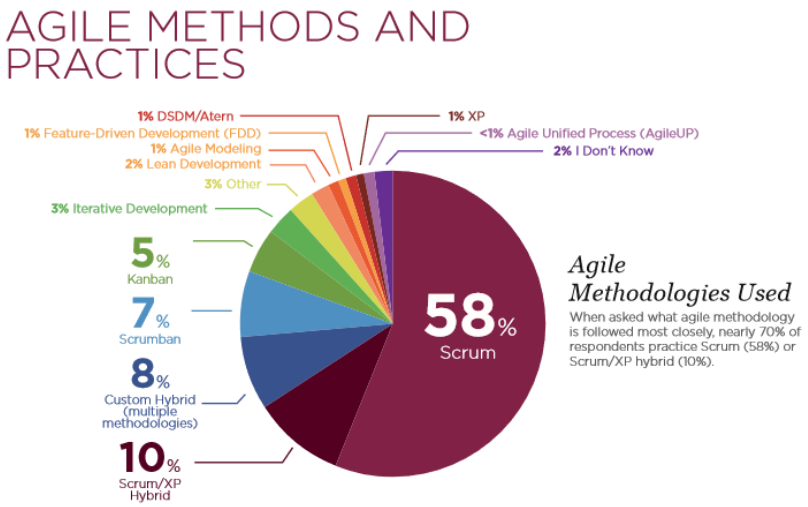


Figura 16. Implementación de metodologías ágiles en VersionOne-Año 2015

Fuente: (VersionOne, 2016)

Para el año 2015, Scrum alcanzó un 58% de implementación sobre las demás metodologías, incluso si tenemos en cuenta las combinaciones de Scrum con XP y Kanban alcanzan una participación del 73% en las implementaciones de VersionOne.

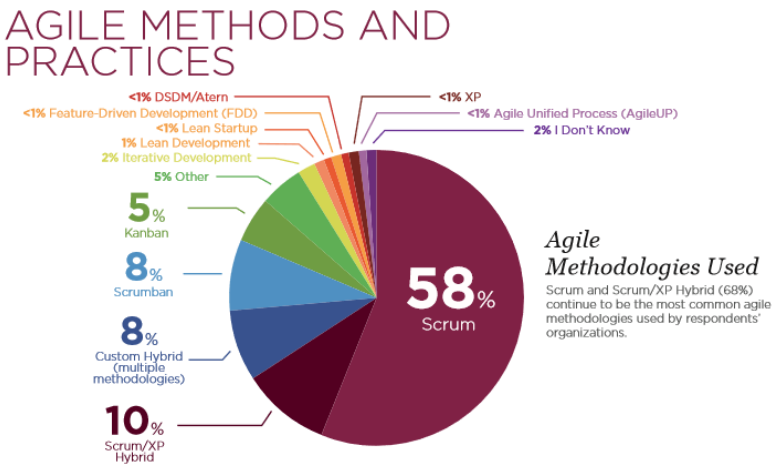


Figura 17. Implementación de metodologías ágiles en VersionOne-Año 2016

Fuente: (VersionOne, 2017)

Para el 2016 las estadísticas del año anterior prácticamente se ratifican. La implementación del Scrum sobre otras metodologías es del 58% y teniendo en cuenta las híbridas o combinadas con XP y con Kanban alcanzan un 76% en implementaciones.

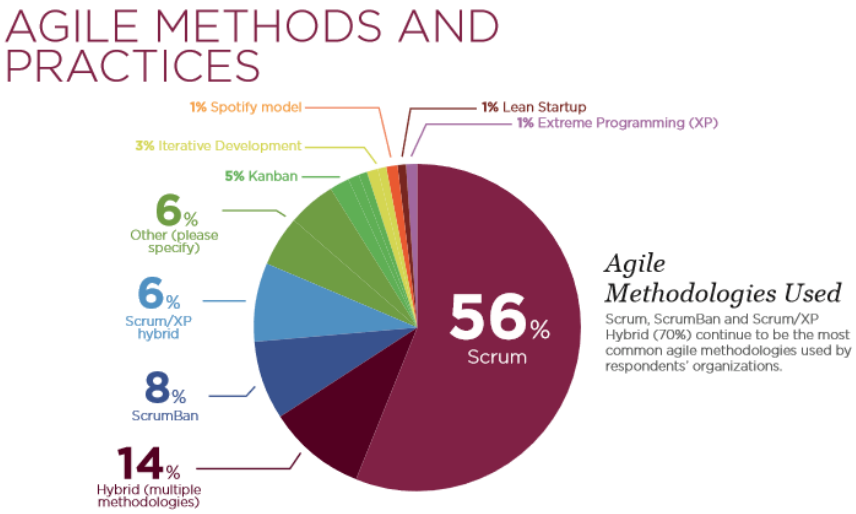


Figura 18. Implementación de metodologías ágiles en VersionOne-Año 2017

Fuente: (VersionOne, 2018)

En el 2017 el porcentaje de implementación de Scrum alcanza un 56%, con un leve descenso en las preferencias sigue siendo la metodología más implementada en VersionOne y teniendo en cuenta las combinaciones de Scrum con otras metodologías como XP y Kanban alcanza un 70% de preferencia.

Scrum ofrece un marco de trabajo flexible, organizado y con autonomía en los equipos de trabajo que permite el desarrollo de un producto de manera ágil. Sin embargo la calidad del producto debe seguir siendo el objetivo principal, entendiendo que la calidad debe enfocarse no solo en el cumplimiento estricto de las necesidades funcionales del cliente sino en entregar un producto seguro, que tenga en cuenta vulnerabilidades para implementar defensas que contrarresten la ocurrencia de ataques informáticos.

6.2. ¿Por qué OWASP?

El trabajo realizado por OWASP es muy importante y plantea lineamientos claros para el fortalecimiento de la seguridad en los aplicativos con arquitectura web.

Los documentos base de OWASP como son el Top 10 de Riesgos informáticos y Controles Proactivos sobre Riesgos, ambos enfocados exclusivamente para aplicativos con esta arquitectura de desarrollo, ofrecen toda la información requerida para aportar a nuestra metodología propuesta los componentes de seguridad requeridos para alcanzar el objetivo planteado.

OWASP (2018) deja claro que:

El OWASP Top 10 - 2017 se basa principalmente en el envío de datos de más de 40 empresas que se especializan en seguridad de aplicaciones y una encuesta de la industria que fue completada por más de 500 personas. Esta información abarca vulnerabilidades recopiladas de cientos de organizaciones y más de 100.000 aplicaciones y APIs del mundo real. Las 10 principales categorías fueron seleccionadas y priorizadas de acuerdo con estos datos de prevalencia, en combinación con estimaciones consensuadas de explotabilidad, detectabilidad e impacto. Uno de los principales objetivos del OWASP Top 10 es educar a los desarrolladores, diseñadores, arquitectos, gerentes y organizaciones sobre las consecuencias de las debilidades más comunes y más importantes de la seguridad de las aplicaciones web. El Top 10 proporciona técnicas básicas para protegerse contra estas áreas con problemas de riesgo alto, y proporciona orientación sobre cómo continuar desde allí.

7. Metodología propuesta (SCOWP)

Se propone una metodología basada en Scrum, aplicable al desarrollo de software donde el producto concreto es un aplicativo web que además de cumplir con los requisitos funcionales y no funcionales exigidos por el cliente, ha sido examinado en una etapa de pruebas contra los riesgos detectados y consignados en el OWASP Top 10 vigente.

Denominada **SCOWP**, por su naturaleza de origen en **SC**rum y su objetivo de seguridad basada en **OW**as**P**.

Formalmente **SCOWP** es una metodología ágil para el desarrollo de software orientado a la web con implementación de controles de seguridad de la información que reduce los efectos negativos de riesgos informáticos teniendo en cuenta la aplicación de las defensas ante vulnerabilidades propias de la arquitectura de desarrollo.

Ha sido creada para proyectos de desarrollo de software que además del cumplimiento de los requisitos funcionales priorice la seguridad de la información como un atributo de calidad del producto final. Está pensada en empresas desarrolladoras de software, departamentos de sistemas y gerentes de proyectos que deban proveer de seguridad extrema el aplicativo a desarrollar. Se resaltan tres componentes estructurales en la metodología **SCOWP**: Elementos (Recurso Humano), Eventos (Reuniones) y Artefactos (Documentos entregables y de apoyo). Para identificar con claridad cuando el texto se refiera a la categorización de cada componente, se establece la siguiente convención:

- Elemento. Primera letra del nombre en mayúscula y todo el nombre en negrilla.

  Ejemplo: …es responsabilidad del **Guía metodológico** el informar…

- Evento. Primera letra del nombre en mayúscula y todo el nombre en negrilla y cursiva

  Ejemplo: ...a la ***Reunión diaria*** pueden asistir …

- Artefacto. Primera letra del nombre en mayúscula y todo el nombre en cursiva

  Ejemplo: ...por ello la *Lista de pendientes* reviste importancia…

Una descripción gráfica de la metodología **SCOWP** y la interacción de todos sus componentes se muestra en la siguiente figura.

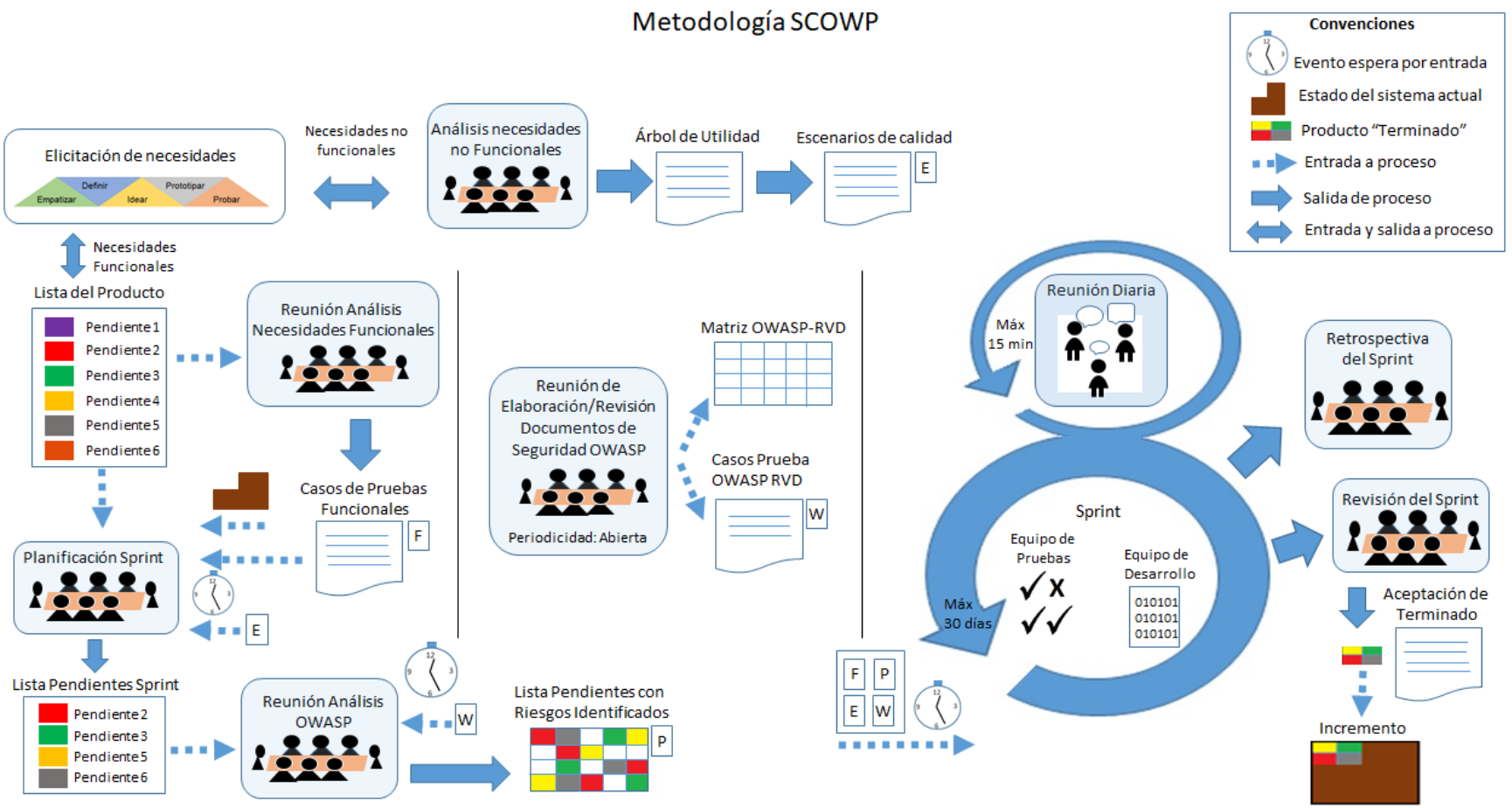


Figura 19. Esquema de la Metodología SCOWP

7.1. Elementos de la metodología propuesta

Se identifican los siguientes componentes en el equipo de trabajo: Dueño del producto, Guía metodológico, Equipo de desarrollo y Equipo de pruebas.

7.1.1. Dueño del producto

Es el representante del cliente ante el proyecto y encargado potencializar el valor del resultado del trabajo del **Equipo de desarrollo**. Es imprescindible que toda la organización respete las decisiones del **Dueño del producto** siendo su mayor empoderamiento el ser responsable en el contenido y en la priorización de la *Lista de producto*, además de aprobar cualquier cambio sobre ésta. También es vital para la aprobación del producto “Terminado” pues su criterio es la voz del cliente que contrata el desarrollo del producto.

6.1.2. Guía metodológico

El **Guía metodológico** para **SCOWP** es un líder que está al servicio de todo el equipo de trabajo, es quien guía a todo el recurso humano involucrado en el desarrollo a cumplir con los lineamientos metodológicos y a solventar todas las situaciones que se presenten entre ellos y que puedan alterar el cumplimiento de objetivos. Como requisito indispensable, debe tener conocimientos de arquitectura de software toda vez que será el líder de desarrollo del producto.

Las funciones del **Guía metodológico** heredadas de Scrum son:

- Procurar en todo momento la consecución de los objetivos del ***Sprint***, y que el alcance y dominio del producto sean de pleno conocimiento y entendimiento por todo el equipo de trabajo.
- Apoyar al **Dueño de producto** a gestionar y ordenar la *Lista de producto* para facilitar el objetivo planteado.
- Dar la confianza y poder al **Equipo de desarrollo** para obtener al final de cada ciclo un producto de alto valor.
- Ser mediador en el **Equipo de desarrollo** de manera que se mantenga el ambiente colaborativo de manera permanente
- Liderar la adopción de la metodología y actuando como facilitador en el control de los cambios ante la organización. Ello permite que el equipo de trabajo consiga una mayor productividad.

Las funciones asignadas por la adopción de los lineamientos de seguridad y del desarrollo de software:

- Elaborar y mantener actualizada la *Matriz OWASP-RVD* conforme el último boletín oficial de OWASP Top 10.
- Dar a conocer la *Matriz OWASP-RVD* a los miembros del **Equipo de desarrollo** y al **Equipo de pruebas**.
- Elaborar el listado de *Casos de pruebas OWASP-RVD* por cada riesgo detectado y consignado en la *Matriz OWASP-RVD.*
- Analizar las prioridades sugeridas por el Dueño del producto y emitir juicios de factibilidad conforme el estado del sistema actual.

Se permite que el **Guía metodológico** encuentre apoyo con asesores en temas de seguridad de la información para llevar a cabo las actividades afines, sin embargo es el directamente responsable ante todo el equipo de trabajo por estas actividades.

El **Guía metodológico** aprueba todos los documentos que implican la preservación de seguridad de la información durante el desarrollo y pruebas aunque la elaboración de algunos de ellos no dependa directamente de él.

6.1.3. Equipo de desarrollo

El **Equipo de desarrollo** lo integran profesionales que deben entregar un producto "Terminado" al final de cada ***Sprint*** y apto para formar parte del sistema en productivo y son ellos los únicos responsables de alcanzar esta meta. Sus principales características son:

- Multifuncionalidad. Tienen formaciones y especialidades diversas y son totalmente autónomos incluso para organizarse. Solo sus integrantes son capaces de convertir elementos de la *Lista de producto* en productos "Terminados" y nadie fuera de ellos puede impartir instrucciones al respecto, manteniendo una estructura compacta donde no existen rangos jerárquicos ni subgrupos en el **Equipo de desarrollo**. Esta condición es permanente. Esto hace que la responsabilidad no se comparta, es de todos así se tengan diversos enfoques o conocimientos especializados.
- No hay directriz sobre el tamaño específico del **Equipo de desarrollo**, sin embargo en **SCOWP** se sugiere que se encuentre entre 4 y 9 miembros.

6.1.4. Equipo de pruebas

Para **SCOWP** es muy importante este elemento, pues es el encargado de ejecutar todas las pruebas y verificar el cumplimiento de los controles en contra de las vulnerabilidades y de las validaciones de los escenarios de calidad. No se establece un límite en el tamaño de su conformación, sin embargo se deja a disposición del **Guía metodológico** la decisión sobre este asunto. Otras funciones asignadas son:

- Ejecutar el listado de *Casos de pruebas OWASP-RVD* por cada riesgo detectado para el pendiente desarrollado en el ***Sprint***.
- Ejecutar la validación de los *Escenarios de calidad* consignados en el formato correspondiente.
- Ejecutar las pruebas de las necesidades funciónales consignados en el documento *Casos de pruebas funcionales*
- Registrar los resultados de las diferentes pruebas y validaciones en el *Formato de resultados de pruebas/validación* del pendiente.
- Comunicar a los resultados a los miembros del **Equipo de desarrollo** y al **Guía metodológico** acerca de las pruebas efectuadas al producto desarrollado y potencialmente “Terminado”
- Aunque no es una función específica del **Equipo de pruebas**, es muy importante que exista un compromiso e interés por mantener un conocimiento actualizado acerca de los riesgos, vulnerabilidades y defensas establecidos por OWASP, de esta manera se afiance el apoyo e interacción el **Guía metodológico.**

7.2. Eventos de la metodología

La metodología **SCOWP** hereda los eventos de Scrum e incluye la *Elicitación de necesidades*, la *Reunión de análisis de riesgos OWASP, Reunión de análisis de necesidades funcionales, Reunión de Elaboración/Revisión de documentos OWASP y Reunión de Análisis de necesidades no funcionales,* las cuales son de vital importancia para la consecución de los objetivos de seguridad del software desarrollado.

7.2.1. Elicitación de necesidades mediante Design Thinking.

El proceso de obtención y definición de las necesidades del cliente para los componentes de software es casi estándar en todos los procesos de desarrollo y se le conoce como levantamiento de información. **SCOWP** sugiere que se llame ***Elicitación de necesidades*** y se trabaje mediante la metodología Design Thinking o en español Pensamiento de Diseño, proponiendo un cambio de paradigma en la forma de pensar al momento de analizar, tal y como lo ejecutaría un diseñador. El Design Thinking tiene su origen en los años 70 en la Universidad de Stanford y ha vuelto a ser implementada con gran éxito para procesos donde el servicio al cliente toma gran valor.

Esta metodología propone cinco etapas en el proceso de entender las necesidades de un cliente: .Empatizar, definir, idear, prototipar y testear.

Lo primordial y por ello es el primer paso es Empatizar con el cliente, crear un vínculo, comprender y entender sus necesidades como si fueran propias (desde el punto de vista del desarrollador). Luego se deben definir tanto los conceptos como las variables para que todos hablen sobre la misma base.

En el idear es propiamente la fase del diseño. “En esta etapa se conciben una gran cantidad de ideas que dan muchas alternativas de donde elegir como posibles soluciones en vez de encontrar una sola mejor solución.” (Hasso Plattner, s.f.)). En la fase de “prototipar” se generan documentos o cualquier otro artefacto que plasme la idea concebida como solución, por lo general se esquematizan de manera gráfica o en diagramas de flujo. Por último se realiza un Testeo (o Prueba) sobre el artefacto prototipado.

¿Por qué sugerir esta metodología para el levantamiento de información?

Porque siendo el trabajo en equipo el principal propósito de nuestra metodología, se requiere que desde el mismo inicio del proyecto el Equipo desarrollador entienda como propias las necesidades del cliente, que se cree desde un principio un vínculo con el cliente y la organización de manera que se construyan las bases para un ambiente colaborativo que favorezca la consecución eficaz de los objetivos.

La salida de este proceso es la identificación de las necesidades funcionales y las no funcionales. Las primeras serán características a tener en cuenta para elaborar la *Lista de producto*. Las necesidades no funcionales serán entrada para la construcción del *Árbol de utilidades* teniéndose en cuenta aquellas cuyos atributos de calidad estén enfocados a la seguridad dentro de los escenarios de calidad de seguridad.

7.2.2. Reunión de Elaboración/Revisión de documentos OWASP

Se establece que el **Guía metodológico** debe convocar a reunión para elaborar y/o revisar cada seis meses dos documentos fundamentales y como resultado es la versión vigente de ellos o su actualización si fuere el caso:

- *Matriz OWASP-RVD*
- C*asos de pruebas OWASP-RVD*

Este evento está inmerso en la metodología pero no se asocia a sprint en particular, de hecho puede realizarse paralela a la ejecución del desarrollo y pruebas y solo se condiciona a la asistencia del **Guía metodológico**, el **Equipo de pruebas** y los asesores que se consideren deban intervenir para conseguir el objetivo de mantener actualizada la información de los documentos anteriormente enunciados. Se sugiere un tiempo máximo de 8 horas para la reunión de elaboración y de máximo 4 horas para la revisión de los documentos existentes. No se determinan fechas para la celebración de estas reuniones, por eso se dejan con periodicidad abierta. El **Guía metodológico** es el responsable de convocarlas cuando lo considere necesario ya sea para elaboración o para actualización de la *Matriz OWASP-RVD* y por consiguiente los *Casos de pruebas OWSP-RVD*.

Los documentos de apoyo por parte del **Guía metodológico** es El OWASP Top 10 de riesgos y el OWASP Controles proactivos, ambos con la última versión vigente.

7.2.3. Reunión de análisis de riesgos OWASP

Una vez finalice la ***Reunión de Planificación del Sprint*** se determina la *Lista de pendientes* que es la entrada para que el **Equipo de desarrollo** registre los atributos para cada componente software a desarrollar (pendiente), identificando cuáles son los riesgos asociados contenidos en la *Matriz OWASP-RVD*

Adicional a lo contemplado en Scrum, se tienen en cuenta los riesgos asociados que se deben evaluar para cada elemento y al igual que los demás, deben al final de cada ***Sprint*** demostrar que han sido atendidos y manejados eficazmente cuando estén "Terminados".

En esta reunión participa el **Equipo de desarrollo** y los interesados o asesores invitados y se sugiere una duración máxima de 4 horas para ***Sprint*** de un mes. Se aplican tiempos proporcionales para estimar la duración de ***Reunión de análisis de riegos OWASP*** cuando el ***Sprint*** sea más corto.

El resultado de esta reunión es la actualización de la **Lista de pendientes,** registrando para cada pendiente el riesgo asociado. Este documento actualizado debe ser de pleno conocimiento del **Equipo de desarrollo** y una herramienta de consulta complementaria a la *Matriz OWASP-RVD* para establecer los controles (Defensas) a tener en cuenta durante la fase de desarrollo de los pendientes. Conocer los riesgos de los pendientes desde la fase de desarrollo permite establecer controles de manera proactiva y minimizar la ocurrencia de errores del software aún en la aplicación de los casos en la etapa de pruebas.

7.2.4. Reunión de análisis de necesidades funcionales

Este evento se sugiere con una duración máxima de 8 horas y asisten el **Equipo de desarrollo,** el **Equipo de Pruebas** o miembros representantes, y el **Dueño del producto** para analizar todas las necesidades funcionales asociadas al proyecto. El **Equipo de desarrollo** y el **Equipo de pruebas** analizan las necesidades funcionales expuestas por el cliente a través del **Dueño del producto o** los interesados invitados. De esta reunión se tiene como resultado el documento *Casos de pruebas de necesidades funcionales*. Para el **Equipo de desarrollo** es fundamental la información registrada para la estimación de los tiempos de los pendientes en la ***Planificación del Sprint*** y para el **Equipo de pruebas** en definir los casos a tener en cuenta en la ***Ejecución de pruebas.***

7.2.5. Reunión de análisis de necesidades no funcionales

Para este evento se sugiere una duración máxima de 8 horas y asisten el **Equipo de Pruebas**, y el **Dueño del producto** para analizar todas las necesidades no funcionales asociadas al proyecto. El **Equipo de Pruebas** puede estar representado por un miembro específico o un subgrupo del mismo y se sugiere la invitación de interesados, preferiblemente usuarios líderes de los procesos en análisis, con el fin de obtener mejores resultados.

Es pertinente aclarar que solo se tendrán en cuenta las utilidades relacionadas a la seguridad de la información. Esta premisa aplica también para los escenarios de

calidad asociados. La razón por la cual se limitan las utilidades de calidad solo a la seguridad es por la naturaleza y enfoque de **SCOWP**.

Como resultado de esta reunión se elabora el *Árbol de utilidad* y *posteriormente los Escenarios de calidad.*

7.2.6. Reunión de Planificación del Sprint

Es una reunión donde principalmente participan el **Dueño del producto**, el **Guía metodológico** y el **Equipo de desarrollo** y se sugiere en una duración máxima de 8 horas para ***Sprint*** regulares de 30 días. Se aplican tiempos proporcionales para reuniones de planificación cuando el ***Sprint*** sea más corto. Es posible la participación de interesados del proyecto o asesores invitados por alguno de los miembros principales de la ***Reunión de planificación del Sprint.***

Los artefactos básicos como entrada a esta reunión son la *Lista de producto, los Escenarios de utilidad*, el estado actual del sistema y los *Casos de pruebas funcionales* de los productos que hacen parte de la ***Lista de producto***. Se evalúan los impactos de las necesidades no funcionales sobre el producto en su fase de desarrollo. En esta reunión el **Equipo de desarrollo** selecciona de la *Lista de producto* las unidades que se van a desarrollar en el ***Sprint*** pues es el único que puede evaluar qué es posible llevar a producto “Terminado” durante el tiempo máximo del ***Sprint***. Un resultado importante de la planificación es el objetivo del sprint, el cual proporciona una guía al **Equipo de desarrollo** sobre el producto a construir.

La salida de la ***Reunión de planificación del Sprint*** es la *Lista de pendientes* del ***Sprint*** que son las actividades a desarrollar tomadas de la *Lista de producto* inicial.

7.2.7. El Sprint

Es un ciclo de trabajo durante el cual se desarrolla un elemento o producto "Terminado" y disponible para entrar a formar parte del sistema actual (*Incremento*). La duración de un ***Sprint*** es máximo de un mes (30 días). Pueden establecerse tiempos más cortos dependiendo de la cantidad de elementos a desarrollarse durante el ***Sprint***. Cada ***Sprint*** contiene otros eventos:

- Reuniones diarias.
- Trabajo de desarrollo y pruebas
- Ejecución de pruebas
- Revisión del Sprint
- Retrospectiva del Sprint

Dado que la naturaleza de metodología ágil permite y es adaptable a cambios, se permiten cambios sin embargo se asume que la ejecución de la ***Planificación del Sprint*** contempla el universo de actividades de manera completa como para que los cambios no logren afectar sustancialmente el tiempo ni el objetivo ya determinados del ***Sprint***. El tiempo del ***Sprint*** no puede modificarse.

¿Y es posible cancelar un ***Sprint***? Aunque es poco común que ocurra, un ***Sprint*** puede ser cancelado y la decisión recae sobre el **Dueño del producto**, quien para ello es autónomo o sigue las recomendaciones de los demás miembros del equipo de trabajo. Si ello ocurre, se debe revisar si el trabajo hasta ese momento puede considerarse "Terminado" y evaluar si pasa a ser un *Incremento* del sistema. Esta decisión recae sobre el **Dueño del producto**.

7.2.8. Reunión diaria

En la metodología **SCOWP** es una reunión de estrictamente 15 minutos máximo al inicio de cada día de trabajo donde el **Equipo de desarrollo** expone lo realizado el día anterior y se estiman las actividades a desarrollar durante la próxima jornada de trabajo a iniciar. Pueden participar otros interesados distintos a los miembros del **Equipo de desarrollo** pero únicamente en calidad de espectadores, no tienen derecho a voz ni voto en dicha reunión.

La reunión debe desarrollarse de pie y sus participantes formarán un círculo de manera que no haya barreras para que todos escuchen lo que el exponente manifiesta. La ausencia de elementos cómodos como las sillas, dinamiza la reunión y aporta significativamente al cumplimiento de un tiempo máximo de 15 minutos.

El uso de ayudas audiovisuales o elementos de apoyo como tableros están permitidos pero deben evitarse teniendo en cuenta que pueden extender la reunión.

El **Guía metodológico** está encargado de verificar el cumplimiento tanto de la reunión como de su objetivo.

7.2.9. Ejecución de pruebas

La ejecución de las pruebas dentro del ***Sprint*** para determinar si un producto está "Terminado" está a cargo del **Equipo de pruebas** el cual podrá invitar interesados al proceso para hacerlo más eficaz. Es permitido y fortalece la actividad el que se incluyan en las pruebas a los líderes funcionales que tendrán a cargo el uso del

pendiente que está a punto de entregarse. El tiempo empleado en las pruebas debe estar incluido en la duración establecida para el Sprint.

SCOWP establece tres tipos de pruebas:

- OWASP: Los casos de pruebas a los riesgos asociados a cada pendiente y se encuentran detallados en el documento *Casos de Pruebas OWASP-RVD.*
- PF: Las pruebas funcionales del pendiente. Corresponden a ejecutar los casos que comprueben que el pendiente se ajusta a las necesidades planteadas por el cliente y están consignados en el documento *Casos de Pruebas Funcionales.*
    - VNNF: Verificación de cumplimiento de las Necesidades no funcionales. Corresponden a los casos que verifican el correcto funcionamiento del pendiente en los escenarios de calidad y están registrados en el documento *Escenarios de calidad*

Para cada pendiente existe un *Formato de resultados de pruebas/validación* el cual consigna el resultado de cada uno de los tipos de pruebas efectuados teniendo en cuenta los casos registrados en la *Lista de pendientes.*

7.2.10. Revisión del Sprint

Se trata de una reunión de, a lo sumo, cuatro horas para ***Sprint*** de 30 días. Se aplican tiempos proporcionales para estimar la revisión cuando el ***Sprint*** sea más corto. Los principales aspectos de este evento son:

- Los asistentes son el **Equipo de desarrollo,** el **Dueño del producto,** el **Guía metodológico** e invitados**.**

- El **Equipo de desarrollo** hace una demostración del producto “Terminado” e incluso exponen los problemas encontrados y la solución implementada.
- Se formaliza el estado de “Terminado” de los pendientes mediante el diligenciamiento del documento *Aceptación de producto “Terminado”* el cual es válido con la firma del **Guía metodológico** y el **Dueño del producto**.
- Si hubiese productos que no se terminaran de la *Lista de Pendientes*, éstos elementos vuelven a la *Lista de Producto*
- La decisión de considerar el producto “Terminado” como un *Incremento* puede tomarse en esta reunión pero es decisión del **Dueño del producto** hacerlo o postergar su determinación. Para efectos contractuales, una vez el producto está “Terminado” se asume cumplido el compromiso por parte del **Equipo de desarrollo.**

En este evento se formaliza el proceso de validación y verificación del software lo cual se ha venido haciendo a lo largo de la fase de desarrollo y pruebas y queda evidenciado en la presentación del producto “Terminado”

7.2.11. Retrospectiva del Sprint

Es una reunión con tiempo máximo sugerido de tres horas para *Sprint* de 30 días. Se aplican tiempos proporcionales para estimar la duración de *Reunión de Retrospectiva* cuando el ***Sprint*** sea más corto.

Participa todo el **Equipo de trabajo** y se trata de evaluarse a sí mismos sobre eventos que hayan ocurrido que permitan crear un plan de mejoras para los *Sprint*

futuros. Se trata más de una puesta en común de sentimientos sobre el trabajo realizado por encima de lo funcional y lo técnico. En esta reunión se puede tomar la decisión de implementar mejoras sobre el trabajo del **Equipo de desarrollo,** incluso es posible que se llegue a replantear los conceptos para considerar un producto "Terminado" según sea conveniente y no entre en conflicto con los estándares adoptados por la organización o del producto mismo.

7.3. Artefactos de la metodología

Los artefactos son los elementos resultantes de reuniones que se constituyen en entrada o salida de los eventos. Estos artefactos también son conocidos como documentos entregables y/o de apoyo y su estructura se puede encontrar en el numeral **9.1. Anexos documentales.** También se consideran artefactos conceptos intangibles como el producto "Terminado" y el Incremento.

7.3.1. Producto "Terminado" versus incremento

Al finalizar el ***Sprint***, los elementos que conformaron la *Lista de pendientes* han debido pasar exitosamente todas las pruebas (Funcionales, OWASP, No Funcionales), de esta manera un producto se considera "Terminado". Esta definición debe ser clara para todo el equipo de trabajo **SCOWP** de manera que no haya malas interpretaciones al momento de evaluar y calificar los productos una vez finalizado cada ***Sprint***.

Dueño del producto es quien decide si un producto "Terminado" se integra o no al sistema actualmente en ambiente productivo. En la metodología **SCOWP**, solo cuando

el **Dueño del producto** decide integrar el producto “Terminado” al sistema en productivo, se considera un *Incremento*.

En consecuencia, el sistema en productivo se constituye de incrementos, mas no necesariamente de todos los productos “Terminados” alcanzados en los ***Sprint***.

7.3.2. Lista de producto

(Ver formato)

Es una lista ordenada y priorizada de todos los elementos necesarios para el producto. Es el **Dueño del producto** el único responsable de la misma. La *Lista de Producto* es un artefacto dinámico, influenciado muchas veces por nuevas necesidades surgidas en la ejecución del ***Sprint*** y otras por requisitos impuestos por el mercado. En todo caso, es susceptible de modificaciones y sobre todo de aumentar aun cuando se avance en el desarrollo del sistema. El **Dueño del producto** es quien autoriza las modificaciones de la *Lista de producto*.

Los elementos de la *Lista de producto* deben estar en un nivel de detalle lo suficientemente claro para que pueda ser desarrollado con una estimación de tiempo lo más acertada posible, de ahí el éxito en la consecución del objetivo del ***Sprint***. El **Equipo de desarrollo** es el responsable de proporcionar todas las estimaciones aunque puede recibir consideraciones a manera de sugerencias del **Dueño del producto.**

7.3.3. Lista de pendientes

(Ver formato)

Son los elementos de la *Lista de producto* seleccionada por el **Equipo de desarrollo** a incluir en el ***Sprint***.

El plan para la entrega de un producto “Terminado” es una predicción hecha por el **Equipo de desarrollo** acerca de qué funcionalidad formará parte del próximo *Incremento* y del trabajo que se requiere (estimaciones) para entregar ese producto “Terminado”. Dado que los elementos son tomados de la *Lista de producto*, se entiende que están lo suficientemente detallados para que la estimación de los tiempos sea lo más acorde a lo real. Así mismo el nivel de detalle ayuda a que los cambios planteados sean fáciles de entender y exponer dentro de la ***Reunión diaria.***

Cuando se requiere nuevo trabajo, el **Equipo de desarrollo** lo adiciona a la *Lista de pendientes* del ***Sprint***. A medida que el trabajo se ejecuta o se completa se va actualizando la estimación de trabajo restante. Cuando algún elemento del plan no se considera necesario, simplemente se elimina. Los cambios sobre la lista de pendientes solo la puede hacer el **Equipo de desarrollo**. La *Lista de pendientes* es una imagen visible en tiempo real del trabajo que el **Equipo de desarrollo** planea llevar a cabo durante el ***Sprint***.

La *Lista de pendientes* de **SCOWP** tiene tres componentes adicionales y corresponden a la identificación de los casos de pruebas que se le deben ejecutar:

- Identificación de riesgos para casos de pruebas OWASP
- Identificación de los casos funcionales
- Identificación de los escenarios de calidad para el caso de la validación de las necesidades no funcionales

7.3.4. Matriz OWASP de Riesgos, Vulnerabilidades y Defensas

(Ver formato)

Reúne el compendio del análisis de los Riesgos, Vulnerabilidades y Defensas contenido en el boletín OWASP Top 10 vigente.

Esta matriz (OWASP-RVD) se debe mantener actualizada con los 10 riesgos más importantes detectados en el último año para los aplicativos en ambiente web y pueden variar dependiendo de nuevas amenazas. Es responsabilidad del **Guía metodológico** su elaboración y actualización, así como su divulgación al **Equipo de desarrollo.**

La Matriz OWASP-RVD se convierte en una herramienta de trabajo del **Equipo de desarrollo** para que determinen los controles (Defensas) que de manera proactiva deben implementar durante la fase de desarrollo del producto.

Tabla 6. *Ejemplo de información que debe contener la Matriz OWASP-RVD*

RIESGO	VULNERABILIDAD (ES)	**DEFENSA (S)**
A1: Inyección	• Invocar consultas dinámicas o no parametrizadas, sin codificar los parámetros de forma adecuada. • Falta de validación de cadenas y concatenación que dan como resultado contenido de datos, comandos o procedimientos almacenados.	• D1: Evitar el uso de Intérpretes y en su lugar usar APIs seguras con interfaz parametrizada. • D2: Validar las entradas digitadas mediante funciones que eviten el ingreso de sentencias SQL • D3: Utilizar funcionalidades SQL como LIMIT y otros controles para evitar fuga masiva de registros • D4: Para cualquier consulta dinámica residual, escape caracteres especiales utilizando la sintaxis de caracteres específica para el intérprete que se trate. • D5: La revisión del código fuente es el mejor método para detectar si las aplicaciones son vulnerables a inyecciones, seguido de cerca por pruebas automatizadas de todos los parámetros, encabezados, URL, cookies, JSON, SOAP y entradas de datos XML.
A2: Pérdida de Autenticación	• Permisibilidad de contraseñas débiles, muy conocidas o por defecto • Permitir ataques de fuerza bruta o ingreso de credenciales automatizadas	• D1: Implementar autenticación multifactor • D2:Evitar el uso de contraseñas por defecto, especialmente los administradores del sistema • D3: Establecer políticas de fuertes establecimiento de contraseñas: Uso de números, Mayúsculas, minúsculas y caracteres especiales obligatoriamente

	• Implementación de procesos débiles de recuperación de contraseña • Empleo de cifrado débiles para el almacenamiento de contraseñas • Niveles de autenticación básicos • Falta de gestión adecuada de rotación de contraseñas	• D4: Establecer límite de intentos fallidos para bloqueo de credenciales • D5: Uso de Captcha para los ingresos persistentes o críticos. • D6: Mecanismos de restablecimiento de contraseñas que involucren mensajería instantánea o cuentas de correo alterno.
A3: Exposición de datos sensibles	• Transmisión de datos e información importante en "texto claro" • Uso de claves criptográficas predeterminadas o reúso de las anteriores • Uso de algoritmos débiles de encripción (MD5, SHA1) • Falta de verificación de los certificados de sitios o de servicios	• D1: Aplicar las políticas regulatorias de Protección de datos • D2: Clasificar la información por su criticidad • D3: Encriptar los datos en la transmisión de la información utilizando protocolos seguros (TLS) • D4: Encriptar la información sensible al almacenar • D5: Evitar el uso de almacenamiento en caché de los datos sensibles • D6: Almacene contraseñas utilizando funciones de hashing adaptables con un factor de trabajo (retraso) además de SALT, como Argon2, script, script o PB-KDF2
A4: Entidades Externas XML (XXE)	• La aceptación de XML directamente, carga XML desde fuentes no confiables o inserta datos no confiables en documentos XML.	• D1: Utilice formatos de datos menos complejos como JSON y evite la serialización de datos confidenciales. • D2: Implemente validación de entrada positiva en el servidor (“lista blanca”), filtrado y sanitización para

	Por último, estos datos son analizados sintácticamente por un procesador XML. • Cualquiera de los procesadores XML utilizados en la aplicación o los servicios web basados en SOAP, poseen habilitadas las definiciones de tipo de documento (DTDs) • La aplicación utiliza SAML para el procesamiento de identidades dentro de la seguridad federada o para propósitos de Single Sign-On (SSO). SAML utiliza XML para garantizar la identidad de los usuarios y puede ser vulnerable. • El uso de SOAP en una versión previa a la 1.2.	prevenir el ingreso de datos dañinos dentro de documentos, cabeceras y nodos XML • D3: Deshabilite las entidades externas de XML y procesamiento DTD en todos los analizadores sintácticos XML en su aplicación, • D4: Revisión del código fuente, incluso empleando herramientas como SAST. • D5: Actualice los procesadores y bibliotecas XML que utilice la aplicación o el sistema subyacente. Utilice validadores de dependencias. Actualice SOAP a la versión 1.2 o superior.
A5: Pérdida de Control de Acceso	• Permitir que la clave primaria se cambie a la de otro usuario, pudiendo ver o editar la cuenta de otra persona. • Posibilidad de manipulación de metadatos como reproducir un token de control de acceso JWT (JSON	• D1: El control de acceso sólo es efectivo si es aplicado del lado del servidor o en Server-less API, donde el atacante no puede modificar la verificación de control de acceso o los metadatos. • D2: Una buena práctica es denegar todos los recursos como política predeterminada.

	Web Token), manipular una cookie o un campo oculto para elevar los privilegios, o abusar de la invalidación de tokens JWT. • Elevación de privilegios. Actuar como un usuario sin iniciar sesión, o actuar como un administrador habiendo iniciado sesión como usuario estándar. • Permitir que la clave primaria se cambie a la de otro usuario, pudiendo ver o editar la cuenta de otra persona. • Elevación de privilegios. Actuar como un usuario sin iniciar sesión, o actuar como un administrador habiendo iniciado sesión como usuario estándar.	• D3: Los modelos de dominio deben hacer cumplir los requisitos exclusivos de los límites de negocio de las aplicaciones. • D4: Deshabilite el listado de directorios del servidor web y asegúrese que los metadatos/fuentes de archivos (por ejemplo de GIT) y copia de seguridad no estén presentes en las carpetas públicas. • D5: Registre errores de control de acceso y alerte a los administradores cuando corresponda. • D6: Limite la tasa de acceso a las APIs para minimizar el daño de herramientas de ataque automatizadas. • D7: Los desarrolladores y el personal de QA deben incluir pruebas de control de acceso en sus pruebas unitarias y de integración.
A6: Configuración de Seguridad Incorrecta	• Falta hardening adecuado en cualquier parte del stack tecnológico, o permisos mal configurados en los servicios de la nube.	• D1: Proceso de fortalecimiento reproducible que agilice y facilite la implementación de otro entorno asegurado. Los entornos de desarrollo, de control de calidad (QA) y de Producción deben configurarse de manera idéntica y con diferentes credenciales para cada entorno. Este proceso puede automatizarse

	• Se encuentran instaladas o habilitadas características innecesarias (ej. puertos, servicios, páginas, cuentas o permisos) • Las cuentas predeterminadas y sus contraseñas siguen activas y sin cambios • El manejo de errores revela a los usuarios trazas de la aplicación u otros mensajes demasiado informativos • Para los sistemas actualizados, las nuevas funciones de seguridad se encuentran desactivadas o no se encuentran configuradas de forma adecuada o segura. • Las configuraciones de seguridad en el servidor de aplicaciones, en el framework de aplicación (ej., Struts, Spring, ASP.NET), bibliotecas o bases de datos no se encuentran especificados con valores seguros. • El software se encuentra desactualizado o posee	para minimizar el esfuerzo requerido para configurar cada nuevo entorno seguro. • D2: Siga un proceso para revisar y actualizar las configuraciones apropiadas de acuerdo a las advertencias de seguridad y siga un proceso de gestión de parches. En particular, revise los permisos de almacenamiento en la nube (por ejemplo, los permisos de buckets S3). • D3: La aplicación debe tener una arquitectura segmentada que proporcione una separación efectiva y segura entre componentes y acceso a terceros, contenedores o grupos de seguridad en la nube (ACLs). • D4: Utilice un proceso automatizado para verificar la efectividad de los ajustes y configuraciones en todos los ambientes. • D5: Implementar una Política Corporativa de Seguridad de la Información • D6: Ejecución de pruebas periódicas que verifiquen el correcto funcionamiento de la configuración de seguridad. De ser posible utilizar herramientas de revisión automática.

	vulnerabilidades conocidas sin atender • Ausencia de una política corporativa de configuración de Seguridad	
A7: Secuencia de comandos de sitios cruzados (XSS)	• XSS Reflejado: Permitir que un segmento del aplicativo o API acepte datos enviados por un usuario u otro procedimiento codificados como parte del HTML o Javascript de salida sin previa validación de los mismos. Esto sucede por no contar con una Política de Seguridad de Contenido. • XSS Almacenado: El almacenamiento de datos recibidos por otro usuario o API que no hayan sido validados. Usualmente es considerado como de riesgo de nivel alto o crítico. • XSS Basado en DOM: La inclusión de datos dinámicos en Javscript o Framework que sean fácilmente controlables por un atacante.	• D1: Utilizar herramientas para generación de frameworks seguros que codifican automáticamente la prevención de XSS como Ruby 3.0 y ReactJS. • D2: Codificar los datos de requerimientos HTTP no confiables en los campos de salida HTML. Esto resuelve el XSS Almacenado y XSS Reflejado. • D3: Aplicar codificación sensitiva al contexto, cuando se modifica el documento en el navegador del cliente, ayuda a prevenir DOM XSS. • D4: Habilitar una Política de Seguridad de Contenido (CSP) es una defensa profunda para la mitigación de vulnerabilidades XSS, asumiendo que no hay otras vulnerabilidades que permitan colocar código malicioso vía inclusión de archivos locales, bibliotecas vulnerables en fuentes conocidas almacenadas en Redes de Distribución de Contenidos (CDN) o localmente.

A8: Deserialización Insegura	Aplicaciones y APIs serán vulnerables si deserializan objetos hostiles o manipulados por un atacante	• D1: El único patrón de arquitectura seguro es no aceptar objetos serializados de fuentes no confiables o utilizar medios de serialización que sólo permitan tipos de datos primitivos. • D2: Implemente verificaciones de integridad tales como firmas digitales en cualquier objeto serializado, con el fin de detectar modificaciones no autorizadas. • D3: Durante la deserialización y antes de la creación del objeto, exija el cumplimiento estricto de verificaciones de tipo de dato, ya que el código normalmente espera un conjunto de clases definibles. Se ha demostrado que se puede pasar por alto esta técnica, por lo que no es aconsejable confiar sólo en ella. • D4: Restrinja y monitoree las conexiones (I/O) de red desde contenedores o servidores que utilizan funcionalidades de deserialización. • D5: Monitoree los procesos de deserialización, alertando si un usuario deserializa constantemente.
A9: Componentes con vulnerabilidades conocidas	• El desconocimiento de las versiones de todos los componentes que utiliza • (tanto del lado del cliente como del servidor). Esto incluye componentes utilizados	• D1: Revisar y remover los componentes de software que no se utilicen. • D2: Tener una herramienta de manejo de versiones de los componentes del software.

	directamente como sus dependencias anidadas. • Contar con un software sin soporte, sin actualizaciones ni parches aplicados. Esto incluye el sistema operativo, servidor web o de aplicaciones, DBMS, APIs y todos los componentes, ambientes de ejecución y bibliotecas. • Falta de seguimiento a los boletines de seguridad de los componentes utilizados en el software.	• D3: Obtener componentes únicamente de orígenes oficiales utilizando canales seguros. Utilizar preferentemente paquetes firmados con el fin de reducir las probabilidades de uso de versiones manipuladas maliciosamente. • D4: Cada organización debe asegurar la existencia de un plan para monitorizar, evaluar y aplicar actualizaciones o cambios de configuraciones durante el ciclo de vida de las aplicaciones.
A10: Registro y Monitoreo Insuficientes	• Falta de registro de eventos auditables tales como los inicios de sesión, fallos en el inicio de sesión, y transacciones de alto valor. • Registro de errores que no sugieren riesgo para el sistema. • Almacenamiento local de los registros, no de manera centralizada en un equipo distinto. • Falta de seguimiento a los umbrales de alertas o tiempos de escalamientos altos.	• D1: Asegúrese de que todos los errores de inicio de sesión, de control de acceso y de validación de entradas de datos del lado del servidor se pueden registrar para identificar cuentas sospechosas. Mantenerlo durante el tiempo suficiente para permitir un eventual análisis forense. • D2: Asegúrese de que las transacciones de alto impacto tengan una pista de auditoría con controles de integridad para prevenir alteraciones o eliminaciones. • D3: Asegúrese que todas las transacciones de alto valor poseen una traza de auditoría con controles de

	• Evidencia que la aplicación no logre detectar alertas y no se gestione su solución.	integridad que permitan detectar su modificación o borrado, tales como una base de datos con permisos de inserción únicamente u similar. • D4: Establezca una monitorización y alerta efectivos de tal manera que las actividades sospechosas sean detectadas y respondidas dentro de períodos de tiempo aceptables.

Fuente de consulta: (OWASP, 2018)

7.3.5. Árbol de utilidad

(Ver formato)

Es el resultado de la ***Reunión de análisis de necesidades no funcionales***. Una vez se identifican los atributos de utilidad, éstos son representados gráficamente de manera que el **Equipo de Pruebas** pueda elaborar posteriormente los *Escenarios de calidad*. Una buena definición de los árboles de calidad permite traducir los objetivos del negocio en escenarios de calidad. Un *Árbol de utilidad* diligenciado es válido únicamente con la aprobación del **Guía metodológico**, el responsable del **Equipo de prueba** y el **Dueño del producto.**

7.3.6. Escenarios de calidad

(Ver formato)

En este formato se registran los escenarios de calidad derivados del *Árbol de Utilidad*. El correcto diligenciamiento permite ejecutar las validaciones de las necesidades no funcionales establecidas por el cliente a través del **Dueño del Producto**. La elaboración de este documento es responsabilidad del **Equipo de pruebas** y debe ser aprobado por su responsable, el **Dueño del producto** y **Guía metodológico**, solo de esta manera podrá ejecutarse la validación de los escenarios registrados. El resultado de las validaciones debe registrarse en el *Formato de resultados de pruebas/validaciones.*

7.3.7. Casos de pruebas OWASP-RVD

(Ver formato)

Es un documento elaborado por el **Guía metodológico** y contiene la casuística de las pruebas a realizar por cada tipo de riesgos consignados en la *Matriz OWASP-RVD.* Es la herramienta de trabajo del **Equipo de pruebas**, quien además de verificar la funcionalidad del producto, debe asegurar que los riesgos de cada elemento de las lista de pendientes han sido contrarrestados con la aplicación de las defensas de rigor. Este documento además de elaborado, es mantenido y actualizado por el **Guía metodológico.** Es válido con la aprobación del **Guía metodológico** y el responsable del **Equipo de pruebas.** El Guía metodológico puede apoyarse en asesores para ejecutar esta actividad pero la responsabilidad de elaboración es completamente de él.

7.3.8. Casos de pruebas funcionales

(Ver formato)

Es un documento donde se consignan los casos que deben validarse por cada pendiente para comprobar la correcta ejecución de las necesidades funcionales asociadas y que se desarrollará en el ***Sprint***. Es responsabilidad del **Equipo de pruebas** elaborar este documento teniendo en cuenta los atributos propios de las necesidades funcionales de cada pendiente. Durante la ***Ejecución de pruebas*** este será la guía de pruebas para cada pendiente funcional y el resultado se registrará en el *Formato de resultados de pruebas*

.

7.3.9. Formato de resultados de pruebas/validación

(Ver formato)

Es un documento que registra los resultados de las pruebas ejecutadas sobre el producto que entrega el **Equipo de desarrollo.**

Solo cuando el *Formato de resultados de pruebas* registra que un pendiente obtuvo resultados exitosos sobre todos los casos de pruebas (OWASP, Funcionales y No Funcionales) identificados y registrados en la *Lista de Pendientes* del ***Sprint***, se da por aceptado el pendiente y se oficializa con la firma del **Guía metodológico** y el (los) responsable (s) del **Equipo de pruebas** en la ejecución de los casos.

El **Dueño del producto** verificará el éxito de todas las pruebas ejecutadas como un requisito antes de aprobar el paso del producto “Terminado” al sistema actual.

7.3.10. Aceptación de producto “Terminado”

(Ver formato)

En la Revisión del ***Sprint*** se realiza la presentación de los pendientes o productos que fueron “Terminados” conforme las necesidades funcionales y no funcionales manifestados por el cliente a través del **Dueño del producto**.

Los productos “Terminados” son formalmente registrados en el documento *Aceptación de producto “Terminado”* y es válido con la firma del **Dueño del producto** y el **Guía metodológico**.

8. Prueba piloto de la metodología SCOWP

La metodología **SCOWP** fue implementada en una prueba piloto en una entidad, integrándola al modelo que emplean actualmente para el desarrollo de sus aplicaciones orientadas a la web. Por acuerdo de confidencialidad suscrito con la entidad no se divulga su nombre pero se anexa la documentación respectiva sobre el piloto realizado en **10.2. Anexos prueba piloto**

8.1. Requerimientos del cliente

Se definió un documento base para las necesidades del cliente y que se buscan cubrir con el software a desarrollar:

Se requiere un aplicativo para la administración de voluntarios a eventos masivos organizados por una entidad. El software debe estar en capacidad de ejecutar los siguientes procesos:

- *Inscripción de voluntarios.*
- *Modificación de datos del voluntario*
- *Convocatoria de voluntarios*
- *Consulta de convocatorias*
- *Registro de asistencia y novedades*
- *Liquidación de honorarios*

Para el piloto que se desarrollará para poner en práctica la metodología SCOWP se toma únicamente el módulo de Inscripción de voluntarios, el cual puede hacerse vía web a través del portal de la organización.

INSCRIPCIÓN DE VOLUNTARIOS

El voluntario debe registrar los siguientes campos con las condiciones que se enuncian:

- *Nacionalidad. Se escoge de una lista predeterminada.*

- *Identificación. Se digita por parte del usuario. Este campo no debe quedar en blanco y se autentica en conjunto con la nacionalidad para el manejo de posibles identificaciones iguales para dos países diferentes.*
- *Nombre. El voluntario debe ingresar el nombre completo. Este campo no debe quedar en blanco.*
- *Correo electrónico. El voluntario debe ingresar su correo electrónico. Este campo no puede quedar en blanco y debe ser único en el sistema porque es uno de los parámetros de acceso a las consultas por parte del voluntario.*
- *Contraseña. Debe tener las siguientes características:*
    - *Al menos 8 caracteres de longitud.*
    - *Contener al menos una letra en mayúscula.*
    - *Contener al menos una letra en minúscula*
    - *Contener al menos un carácter especial*
    - *Se debe restringir el uso de secuencia numérica consecutiva.*
    - *La contraseña no puede contener ni el identificador del correo, ninguno de los nombres ni ninguno de los apellidos.*
    - *No podrá contener el carácter espacio.*
- *Nivel de estudios. El voluntario debe registrar su nivel de estudios y lo escoge dentro de una lista predeterminada.(Bachiller, Universitario, Posgrado)*
- *Idioma. El voluntario debe registrar el o los idioma (s) que domina. Se escogerá dentro de una lista predeterminada, (Inglés, Español, Francés, Alemán, Italiano, Portugués, etc)*
- *Preferencia del voluntario. El voluntario podrá escoger la categoría de los eventos a los que prefiere postularse. (Deportes, Salud, Recreación, Todos)*
- *Dirección de residencia. Dirección de la residencia del voluntario.*
- *Teléfono. Teléfono celular o fijo de contacto.*
- *Municipio de residencia. Ciudad o municipio donde reside permanentemente el voluntario.*

El proceso finaliza con la grabación de la información contenida en los campos de la plantilla y tanto el almacenamiento como la transmisión deben proveer medidas de seguridad de la información sensible.

Se implementó también una versión App para móviles y se obtiene de la PlayStore con el nombre de Voluntarios Barranquilla.

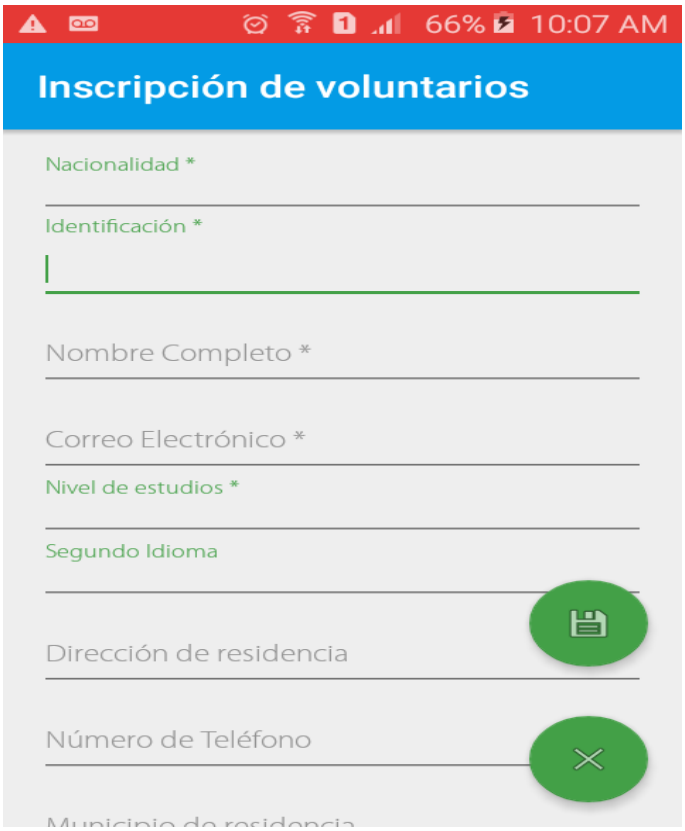


Figura 20. Imagen de aplicativo piloto-Versión App

8.2. Resultados de la prueba piloto

La implementación de la metodología **SCOWP** en el piloto de desarrollo dejó ver lecciones aprendidas dentro de las expectativas planteadas en la naturaleza de Scrum como metodología base. En primer lugar contar con información de tipo analítico como

los riesgos y controles (defensas) propuestos por OWASP delinean un camino hacia el fortalecimiento de la seguridad de la información y haber incluido ese análisis en este trabajo fue un acierto en pro de la consecución del objetivo planteado. En segundo lugar el poner en práctica los pasos de manera rigurosa permitió afinar la propuesta, tanto en lo metodológico como en lo documental. En algún momento del piloto se evidenciaron lagunas en las responsabilidades y roles que fueron inmediatamente corregidas en cabeza de los responsables funcionales según el caso y se rediseñaron formatos que permitieron que la documentación facilitara el registro de información y no que imprimiera complejidad al proceso.

9. Conclusiones

Los ataques informáticos que afectan los aplicativos con arquitectura web son cada vez más frecuentes muy a pesar de los esfuerzos de los profesionales de TI en generar medidas que contrarresten el accionar de los ciberdelincuentes. Cuando un software web pasa al ambiente productivo, aunque establezca condiciones para acceder a su ejecución y servicio, está expuesto a que cualquier individuo pretenda accederlo desde cualquier punto geográfico del planeta. Las empresas que emplean metodologías ágiles para desarrollar software orientado a la web deben implementar controles que incrementen la seguridad en sus productos, tanto del código mismo como de la información que almacenan y manejan, como un atributo de calidad diferenciador dentro del mercado de TI.

Como demuestra este trabajo de investigación, es factible fortalecer éstas metodologías con buenas prácticas de seguridad de la información y aplicar resultados de estudios de alto nivel con formulación de defensas ante riesgos de frecuente ocurrencia de manera que se tengan en cuenta los controles contra las vulnerabilidades desde las etapas iniciales análisis y diseño, implementarlos en la fase de desarrollo y ejecutar un minucioso plan de pruebas para obtener un producto final que cumpla tanto con las necesidades funcionales planteadas por el cliente, como con las necesidades no funcionales ofrecidas por el compromiso del desarrollador de entregar un producto con un atributo de seguridad que incrementa su calidad.

Precisamente por ser **SCOWP** una propuesta de metodología ágil, la documentación contenida es la básica y suficiente para mantener agilidad sin que se pierda la

trazabilidad dentro del desarrollo e integración de los productos en un proyecto que si bien es modular requiere de una huella para el proceso de mantenimiento una vez cada componente entre a ambiente productivo.

Se demostró que **SCOWP** cumple a cabalidad con el objetivo planteado y articula exitosamente la metodología ágil Scrum con los lineamientos de seguridad de OWASP ofreciendo a las empresas desarrolladoras de software comprometidas con la seguridad de la información, una nueva metodología para desarrollo de aplicativos orientado a la web con un nivel alto de seguridad informática como atributo de calidad del producto final.

Con respecto al costo de implementación de **SCOWP** en una organización (inquietud frecuente de parte del personal administrativo de las empresas), es importante resaltar que la metodología es flexible en el tamaño de los equipos de trabajo. El costo de implementación está relacionado con la decisión de sumar miembros al **Equipo de desarrollo** y al **Equipo de pruebas** dependiendo de la cantidad de pendientes que se requieran desarrollar en un ***Sprint***, sin embargo es importante recordar que éstos costos son muchísimo menores si los comparamos a los asociados a la corrección de errores en la fase operación productiva del software y casi ínfimos si los comparamos con el potencial impacto económico que puede ocasionar un ataque informático ante la ausencia de controles proactivos a vulnerabilidades web que hayan podido ser solventadas en el análisis, diseño y desarrollo del aplicativo.

Línea de investigación propuesta para trabajos futuros

Se propone como trabajos futuros derivados de esta tesis, extender el análisis para la formulación de una metodología que fortalezca la seguridad de los servicios de tecnología ofrecidos en la nube como lo son, entre otros:

- Software como servicio (SaaS)
- Infraestructura como servicio (IaaS)
- Seguridad como servicio (SECaaS)

Todos estos servicios por ser ofrecidos en la nube afrontan los mismos riesgos web que presentan el software orientado a esta arquitectura.

10. Referencias

Adasoft, 2017. *30 Segundos*. 5 cifras y estadísticas sobre ciberseguridad. Recuperado de https://adasoft.es/5-cifras-estadisticas-de-ciber-seguridad-para-2017/

Aguilera, G., y De las Salas, D. (2014).*Beneficios Tributarios en Colombia por la exportación de servicios de software*. (Trabajo Grado de Especialización) Recuperado de http://expeditiorepositorio.utadeo.edu.co/bitstream/handle/20.500.12010/1509/T122.pdf?sequence=1&isAllowed=y

Alfaro, O., s.f. Project Management Agile. Recuperado de http://www.alfarosolis.com/content/PDFs/IF7100/Semana10/Calidad.pdf

Alfonso, K. (22 de Septiembre de 2017). Industria del software y tecnologías de la información aporta 1,6% del PIB anual. *La República.* Recuperado de https://www.larepublica.co/economia/industria-del-software-y-tecnologias-de-la-informacion-aporta-16-del-pib-anual-2551102

Alfonso, K. (9 de Enero de 2018). Industria del software y tecnologías de la información aporta 1,6% del PIB anual. *La República.* Recuperado de https://www.larepublica.co/economia/industria-de-software-colombiana-espera-lograr-ventas-por-18-billones-durante-2018-2586708

Almunia, P. (2004). Ciclo de vida en el Desarrollo de Software, segunda parte. [Mensaje de un blog]. Recuperado de https://www.iedge.eu/pablo-almunia-ciclo-de-vida-en-el-desarrollo-de-software-segunda-parte

Alvarez, I. (2012).Departamento de electrónica y ciencias de la computación. Cali, Colombia: Pontificia Universidad Javeriana Cali. Recuperado de http://cic.puj.edu.co/wiki/lib/exe/fetch.php?media=materias:sg07.p02.scrum.pdf

Bermejo, M. (2010). *Cámara de comercio de Armenia.* Armenia, Colombia: Cámara de Comercio de Armenia. Recuperado de http://www.camaraarmenia.org.co/files/Produccion_multimedia_(Modulo_4).pdf

Britto, O., & Dapozo, G. (2011). *Universidad Nacional del Nordeste.* Agentina: Universidad Nacional del Nordeste. Recuperado de http://www.unne.edu.ar/unnevieja/investigacion/com2011/CE-Web/CE-047.pdf

Canós, J, Letelier, P., y Panadés, M. (12 de Noviembre de 2003). *Laboratorio Docente de Computación*. Venezuela: Universidad Simón Bolívar. Recuperado de https://ldc.usb.ve/~abianc/materias/ci4713/actasMetAgiles.pdf

Díaz, J., Banchoff, C., Rodríguez, A., y Soria, V. (2009). *Sedici: Repositorio Institucional de la UNLP.* Argentina: Universidad Nacional de la Plata. Recuperado de http://sedici.unlp.edu.ar/bitstream/handle/10915/21017/Documento_completo.pdf?sequence=1

Dimtec. (7 de Septiembre de 2018). *Digital Media Technologies* Ataques a aplicaciones web crecen 30 % en T3-2017.. Recuperado de http://www.dimtec.com/blog/ver/ataques-a-aplicaciones-web-crecen-30-en-t3-2017-akamai

Dinero (Octubre 27 de 2015). La industria de software 'criolla' dio un salto de calidad para conquistar el mercado. *Dinero*. Recuperado de https://www.dinero.com/pais/articulo/progreso-industria-del-software-colombiana/215210

Drake, J. y López, P. (2009). *Ingeniería de software y tiempo real.* España: Universidad de Cantabria. Recuperado de https://www.ctr.unican.es/asignaturas/Ingenieria_Software_4_F/Doc/M7_09_VerificacionValidacion-2011.pdf

El Tiempo. (27 de Septiembre de 2017). A diario se registran 542.465 ataques informáticos en Colombia. El Tiempo. Recuperado de https://www.eltiempo.com/tecnosfera/novedades-tecnologia/informe-sobre-ataques-informaticos-en-colombia-y-al-sector-financiero-135370

EXCLE. (2 de Mayo de 2018). *Soluciones Biométricas*. La biometría llega a los Captchas. Recuperado de http://www.ex-cle.com/la-biometria-llega-a-los-captchas/

Fedesoft.(2015).*Informe de Caracterizaciónde la industria del software colombiano*. Recuperado de http://fedesoft.org/noticias-fedesoft/disponible-estudio-de-caracterizacion-de-la-industria-del-software-colombiano/

Fonseca, E. (2016). *Evaluación de la efectividad de las técnicas de pruebas de software estructurales y funcionales mediante replicación experimental, caso práctico ESPE sede Latacunga*. (Tesis de Maestría) Recuperado de http://repositorio.espe.edu.ec/bitstream/21000/13328/1/T-ESPEL-MAS-0032.pdf

García, M (Julio de 2015). *Estudio comparativo entre las metodologías ágiles y las metodologías tradicionales para la gestión de proyectos de software.* (Tesis de Maestría) Recuperado de http://digibuo.uniovi.es/dspace/bitstream/10651/32457/6/TFMMlJGarciaRodriguezRUO.pdf

Grados, L., y La Serna, N. (11 de Abril de 2015). Desarrollo de un marco metodológico del proceso de verificación y validación de software para pequeñas y medianas empresas. *Industrial Data, 18* (2), 145-154.

Guerrero, J., Jiménez, R., Muñoz, J., Zambrano, A. y Ojeda, G. (2017). Aspectos fundamentales en la construcción de páginas web seguras basados en OWASP. *Boletín Informativo CEI 4(2)*, 216-218.

Hasso Plattner, Institute of Design at Standfor. (s.f.). Guía del Proceso creative: Una introducción al Design Thinking. Recuperado de https://dschool-old.stanford.edu/sandbox/groups/designresources/wiki/31fbd/attachments/027aa/GU%C3%8DA%20DEL%20PROCESO%20CREATIVO.pdf?sessionID=8af88fee76ecd1fb7879c915073461486c425622

Joskowicz, J. (1 de Febrero de 2008). *IIE: Instituto de Ingenería Eléctrica*. Uruguay: Universidad de la República. Recuperado de http://iie.fing.edu.uy/~josej/docs/XP%20-%20Jose%20Joskowicz.pdf

Letelier, P., y Panadés, M.. (15 de Enero de 2006). *Ciencia y Técnica Administrativa*. Argentina. Recuperado de http://www.cyta.com.ar/ta0502/b_v5n2a1.htm

López, F. (15 de Enero de 2015). *Metodologías para el desarrollo de software seguro.* (Tesis de Pregrado). Recuperado de https://upcommons.upc.edu/bitstream/handle/2099.1/24902/103275.pdf?sequence=1

Matei, A. (2015). *Guía para el desarrollo de software seguro.* (Tesis de Pregrado). Recuperado de http://oa.upm.es/34770/1/PFC_ADRIANA_MATEI.pdf

Microsoft,s.f. Security Development Lifecycle.. What is SDL? Recuperado de https://www.microsoft.com/en-us/SDL

Microsoft,s.f. Technet. Ciclo de vida de desarrollo seguro de software. Recuperado de https://social.technet.microsoft.com/wiki/contents/articles/36676.ciclo-de-vida-de-desarrollo-seguro-de-software-es-es.aspx

Milano, P. (12 de Septiembre de 2007). *Seguridad en el ciclo de vida del desarrollo de software.* Cybsec: Security Systems. Recuperado de http://www.cybsec.com/upload/cybsec_Tendencias2007_Seguridad_SDLC.pdf

Molina, B., Vite, H., y Dávila, J. (Julio de 2018). Metodologías ágiles frente a las tradicionales en el proceso de desarrollo de software. *Revista Espirales, 2 (17), 114-120. .* Recuperado de http://revistaespirales.com/index.php/es/article/view/269/225

Muñoz, M., y García, L. (2017). *Análisis de riesgos y prototipo de una página web mediante autenticación CAPTCHA.* (Trabajo de Grado de Especialización). Recuperado de http://190.131.241.186/bitstream/handle/10823/1003/Documento%20Final.pdf?sequence=1

Muñoz, M., Mejia, J. y Corona, B. (15 de Diciembre de 20162016). Hacia la evaluación de la implementación y uso de metodologías ágiles en las pymes: Un análisis de herramientas de evaluación de metodologías ágiles. *International Journal of Information Systems and Software Engineering for Big Companies (IJISEBC) , 3(2),* 75-82. Recuperado de http://uajournals.com/ojs/index.php/ijisebc/article/view/176

Navarro, A.; Fernández, J., y Morales, J. (20 de Septiembre 2013). Revisión de metodologías ágiles para el desarrollo de software. *PROSPECTIVA, 11* (2), 30-39.

Navarro, M., Moreno, M., Aranda, J., Parra, L., Rueda, J., y Pantano, J. (Abril de 2017). *Sedici: Repositorio Institucional de la UN LP.* Argentina: Universidad Nacional de la Plata. Recuperado de http://sedici.unlp.edu.ar/handle/10915/62179

Orjuela, A., y Rojas, M. (24 de Mayo de 2008). *Bdigital: Repositorio Institucional UN.* Colombia: Universidad Nacional de Colombia. Recuperado de http://bdigital.unal.edu.co/15430/1/10037-18216-1-PB.pdf

OWASP (2018). *Top 10 Controles Proactivos 2018*. Recuperado de https://www.owasp.org/index.php/OWASP_Proactive_Controls#5:_Implement_Identity_and_Authentication_Controls

OWASP. (2018). *10 Critical security areas that software developers must be aware to..* Recuperado de https://www.owasp.org/images/b/bc/OWASP_Top_10_Proactive_Controls_V3.pdf

OWASP. (2018). *Los diez riesgos más críticos en aplicaciones web.* Recuperado de https://www.owasp.org/images/5/5e/OWASP-Top-10-2017-es.pdf

Padilla L. (Mayo 5 de 2010). Manufactura esbelta/ágil. *Ingeniería Primero, 15*(1), 64-69. Recuperado de http://files.udesprocesos.webnode.es/200000028-6743f683e7/manufactura%20esbelta%20toyota.pdf

Pérez, B. (2003). *INCO-Instituto de Computación.* Uruguay: Universidad de la República. Recuperado de https://www.fing.edu.uy/inco/pedeciba/bibliote/reptec/TR0417.pdf

Portafolio. (06 de Agosto de 2015). El sector software vende 9,3 billones de pesos al año. Portafolio. Recuperado de http://www.portafolio.co/negocios/empresas/sector-software-vende-billones-pesos-ano-24860

Portafolio. (21 de Mayo de 2018). Industria del 'software' crecería 19% en el 2018. *Portafolio*. Recuperado de http://www.portafolio.co/negocios/industria-del-software-creceria-19-en-el-2018-517332

Proyectos Agiles (s.f.). *Qué es Scrum?.* Recuperado de https://proyectosagiles.org/que-es-scrum/

Rodriguez, P. (Septiembre de 2008). *Estudio de la aplicación de las metodologías ágiles para la evolución de productos software*.(Tesis de Maestría) Recuperado de http://oa.upm.es/1939/1/TESIS_MASTER_PILAR_RODRIGUEZ_GONZALEZ.pdf

Shwaber, K. y Sutherland, J. (Noviembre de 2017). *Scrum Guides.* Organización SCRUM. Recuperado de https://www.scrumguides.org/docs/scrumguide/v2017/2017-Scrum-Guide-Spanish-SouthAmerican.pdf

Tarazona, C. (2007). *Portal de Revistas*. Colombia: Universidad Externado. Amenazas Informática y seguridad de la información. Recuperado de https://revistas.uexternado.edu.co/index.php/derpen/article/download/965/915/

Trigas, M. (2012). *Gestión de Proyectos Informáticos: Metodología Scrum.* Recuperado de http://openaccess.uoc.edu/webapps/o2/bitstream/10609/17885/1/mtrigasTFC0612memoria.pdf

UNP.(2015) *Análisis del sector Tecnologías de la Información y Comunicación e Informática.* Recuperado de https://www.unp.gov.co/la-unp/Documents/DA_PROCESO_15-13-3937756_211001041_15096259.pdf

Varas, M. (2000). *Gestión de Proyectos de Desarrollo de Software.* Concepción, Chile: Academia.edu. Recuperado de https://s3.amazonaws.com/academia.edu.documents/32026338/importante.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1538750967&Signature=Lp5CJ4orVP5PsyJwVSHN7Qyied4%3D&response-content-disposition=inline%3B%20filename%3DImportante.pdf

VersionOne (13 de diciembre de 2016). *10th Annual State of Agile Report.* Recuperado de https://explore.versionone.com/state-of-agile/versionone-10th-annual-state-of-agile-report-2

VersionOne. (06 de abril 2017). *11th Annual State Agile Report*. Recuperado de https://explore.versionone.com/state-of-agile/versionone-11th-annual-state-of-agile-report-2

VersionOne. (09 de Abril de 2018). *12th Annual State Agile Report*. Recuperado de https://explore.versionone.com/state-of-agile/versionone-12th-annual-state-of-agile-report

Villalba, A. y Corchado, J. (2017). *DIALNET*. España: Universidad de La Rioja. Recuperado de https://dialnet.unirioja.es/servlet/articulo?codigo=6115622

Zamora, J. (2011). *Análisis de los procesos de Verificación y Validacón en las organizaciones software.* (Tesis de Pregrado). Recuperado de https://orff.uc3m.es/bitstream/handle/10016/12880/PFC_Jorge_Zamora_Hernandez.pdf;jsessionid=BE79E67CF616CD2EDD5EFB8EE716FCBF?sequence=1

11. Anexos

11.1. Anexos Documentales

11.1.1. Lista de producto

Scowp

Metodología ágil

Lista de Producto

Espacio Reservado para Logo Institucional

Nombre del proyecto :

Dueño del producto :

Id. Prod.	Descripción del Producto	Prior.	T. Estim.	Sprint	Estado	Adiciones/Cambios

Firmas:

Metodología ágil para desarrollo de software web seguro

11.1.2. Lista de pendiente

Lista de Pendientes

Espacio Reservado para Logo Institucional

Id. Producto: _____
Sprint _____
Fecha: *dd/mm/aaaa*

Id. Pend.	Descripción del pendiente	Responsable	Prior.	T. Est.	Estado	Riesgos OWASP	Pruebas Funcionales	Escenarios de Nec. No Funcionales

Firmas:

Metodología ágil para desarrollo de software web seguro

11.1.3. Matriz OWASP-RVD

Scowp

Metodología ágil

Matriz OWASP-RVD

Espacio Reservado para Logo Institucional

Fecha Elaboración Matriz:	*(dd/mm/aaaa)*
Versión OWASP Top 10:	*(--- Año ---)*
Versión OWASP Controles:	*(--- Año ---)*

RIESGO	VULNERABILIDAD (ES)	DEFENSA (S)
A1: *Riesgo 1*		
A2: *Riesgo 2*		
A3: *Riesgo 3*		
A4: *Riesgo 4*		
A5: *Riesgo 5*		
A6: *Riesgo 6*		
A7: *Riesgo 7*		
A8: *Riesgo 8*		
A9: *Riesgo 9*		
A10: *Riesgo 10*		

Firma de Guía metodológico : _____

11.1.4. Árbol de Utilidad

Árbol de Utilidad

Espacio Reservado para Logo Institucional

Fecha: *dd/mm/aaaa*
Id. Producto:
Descripción del Producto
Id. Utilidad : **Nombre:**

Guía Metodológico: Firma:

Resp. Equipo Pruebas: Firma:

Dueño del producto: Firma:

Metodología ágil para desarrollo de software web seguro

11.1.5. Escenarios de calidad

Escenarios de calidad

Espacio Reservado para Logo Institucional

Id. Producto _____

Descripción Producto _____

Id. Atributo de ultilidad _____

Id. Escenario de calidad	
Descripción escenario:	
Generador del estímulo	
Estímulo	
Entorno	
Componente	
Respuesta	
Medida de la respuesta	

Id. Escenario de calidad	
Descripción escenario:	
Generador del estímulo	
Estímulo	
Entorno	
Componente	
Respuesta	
Medida de la respuesta	

Id. Escenario de calidad	
Descripción escenario:	
Generador del estímulo	
Estímulo	
Entorno	
Componente	
Respuesta	
Medida de la respuesta	

Guía Metodológico: _____ Firma: _____

Resp. Equipo Pruebas: _____ Firma: _____

Dueño del producto: _____ Firma: _____

11.1.6. Casos de Pruebas OWASP-RVD

Casos de pruebas OWASP-RVD

Espacio Reservado para Logo Institucional

Año de Matriz OWASP-RVD: _____
Id. Riesgo OWASP: _____

Id. Caso de prueba: _____

Descripción de la actividad de prueba

Condiciones previas del sistema

Respuesta esperada del sistema

Id. Caso de prueba:

Descripción de la actividad de prueba

Condiciones previas del sistema

Respuesta esperada del sistema

Id. Caso de prueba:

Descripción de la actividad de prueba

Condiciones previas del sistema

Respuesta esperada del sistema

Firma Guia metodológico: _____

Firma Resp. Equipo Pruebas _____

Metodología ágil para desarrollo de software web seguro

11.1.7. Casos de pruebas funcionales

Casos de pruebas funcionales

Espacio Reservado para Logo Institucional

Id. Pendiente _____
Sprint _____

Id. P. Funcional: _____

Descripción de la actividad de prueba

Condiciones previas del sistema

Respuesta esperada del sistema

Id. P. Funcional: _____

Descripción de la actividad de prueba

Condiciones previas del sistema

Respuesta esperada del sistema

Id. P. Funcional: _____

Descripción de la actividad de prueba

Condiciones previas del sistema

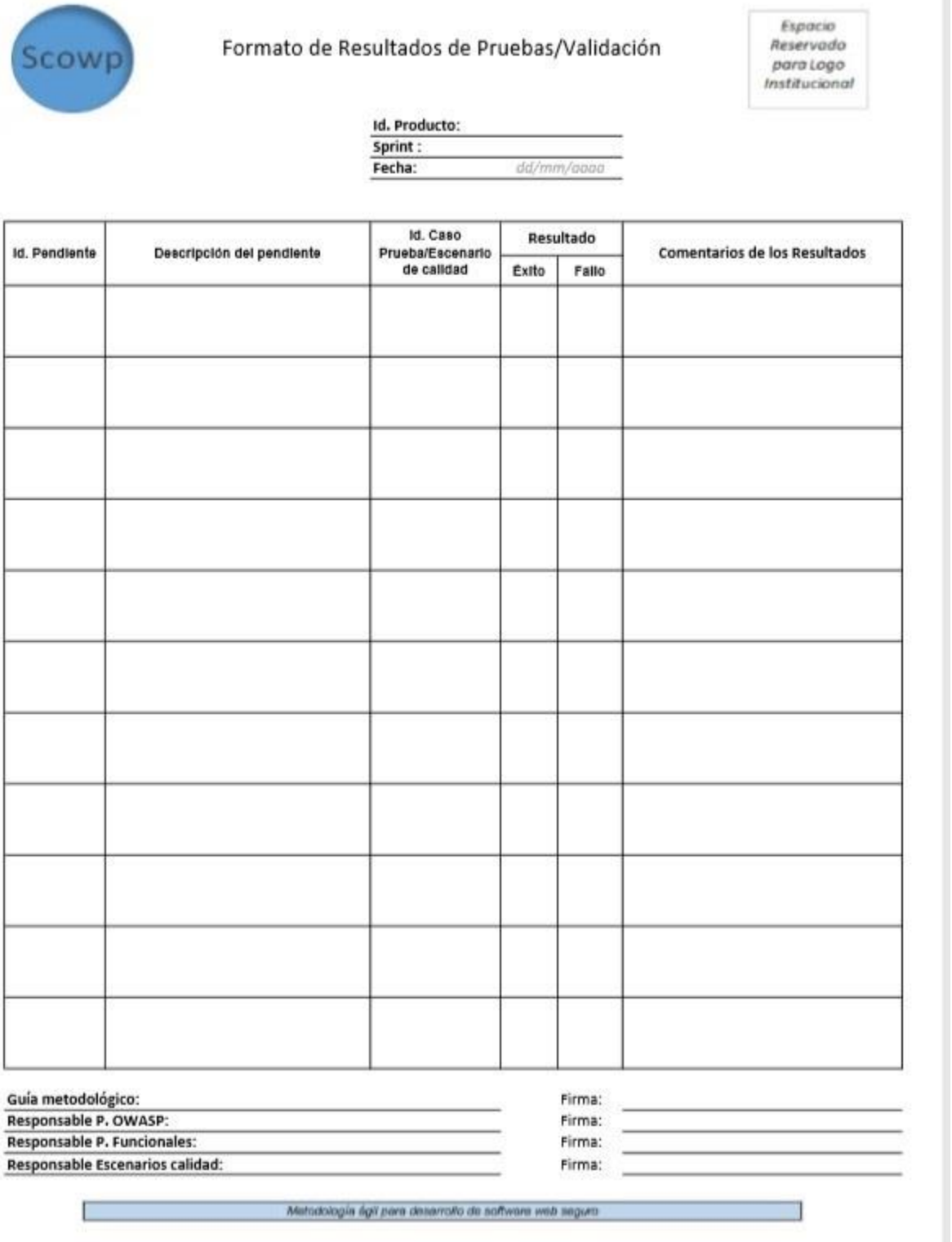
Respuesta esperada del sistema

Firma Guía metodológico: _____

Firma Resp. Pr. Func. _____

Metodología ágil para desarrollo de software web seguro

11.1.8. Formato de resultados de pruebas/validación



Scowp

Formato de Resultados de Pruebas/Validación

Espacio Reservado para Logo Institucional

Id. Producto:
Sprint :
Fecha: *dd/mm/aaaa*

Id. Pendiente	Descripción del pendiente	Id. Caso Prueba/Escenario de calidad	Resultado		Comentarios de los Resultados
			Éxito	Fallo	

Guía metodológico: **Firma:**
Responsable P. OWASP: **Firma:**
Responsable P. Funcionales: **Firma:**
Responsable Escenarios calidad: **Firma:**

Metodología ágil para desarrollo de software web seguro

11.1.9. Aceptación de producto “Terminado”

Aceptación de producto "Terminado"

Espacio Reservado para Logo Institucional

Fecha: *dd/mm/aaaa*
Año/Sprint /

Dentro de la revisión del Sprint a quienes asistieron:

se determinó que los siguientes productos fueron aprobados y se consideran "Terminados" y listos para ser Incrementos del sistema actual:

Id. Producto	Descripción del producto

Guía Metodológico: _____ Firma: _____

Dueño del producto: _____ Firma: _____

11.2. Anexos prueba piloto

10.2.1. Lista de Producto

Scowp

Lista de Producto

Espacio Reservado para Logo Institucional

Nombre del proyecto : Administración de base de voluntarios
Dueño del producto : Atenógenes Miranda

Id. Prod.	Descripción del Producto	Prior.	T. Estim. (dias)	Sprint	Estado	Adiciones/Cambios
VOL-INS	Inscripción de voluntarios al sistema	1	3	1	Terminado	
VOL-VOC	Convocatoria de voluntarios	3	3	2	Pendiente	
VOL-SUL	Consulta de voluntarios	2	2	2	Pendiente	
VOL-NOV	Registro de asistencia y novedades	2	4	3	Pendiente	
VOL-LIQ	Liquidación de honorarios de voluntarios	2	3	3	Pendiente	

Firmas: (GM) (DUEÑO DE PRODUCTO)

Metodología ágil para desarrollo de software web seguro

10.2.2. Lista de Pendiente

Lista de Pendientes

Espacio Reservado para Logo Institucional

Id. Producto: VOL-INS Administración de base de voluntarios
Sprint 1
Fecha: 16/10/2018

Id. Pend.	Descripción del pendiente	Responsable	Prior.	T. Est. (d)	Estado	Riesgos OWASP	Pruebas Funcionales	Escenarios de Nec. No Funcionales
VOL_INS_10	Ingresar Dirección de residencia	CM	1	0,1	Terminado	A1-D1-01	VOL-INS-NBLAN	
VOL_INS_11	Ingresar municipio de residencia	CM	1	0,1	Terminado	A1-D1-01	VOL-INS-NBLAN	
VOL_INS_12	Grabar información	CM	1	0,3	Terminado	A3-D4-01	VOL-INS-NBLAN	SEG-INT-01
VOL_INS_13	Cancelar ingreso de voluntario	CM	1	0,1	Terminado		VOL-INS-CANC	

Firmas: (GM) (DUEÑO DE PRODUCTO)

Lista de Pendientes

Espacio Reservado para Logo Institucional

Id. Producto: VOL-INS Administración de base de voluntarios
Sprint 1
Fecha: 16/10/2018

Id. Pend.	Descripción del pendiente	Responsable	Prior.	T. Est. (d)	Estado	Riesgos OWASP	Pruebas Funcionales	Escenarios de Nec. No Funcionales
VOL_INS_01	Ingresar la nacionalidad	CM	1	0,1	Terminado		VOL-INS-NBLAN	
VOL_INS_02	Ingresar identifiación	CM	1	0,2	Terminado	A1-D1-01	VOL-INS-IDEN VOL-INS-NBLAN	
VOL_INS_03	Ingresar el nombre completo del voluntario	CM	1	0,1	Terminado	A1-D1-01	VOL-INS-NBLAN	
VOL_INS_04	Ingresar ecorreo electrónico del voluntario	CM	1	0,1	Terminado	A1-D1-01	VOL-INS-NBLAN VOL-INS-EMAIL	
VOL_INS_05	Ingresar nivel de estudios del voluntario	CM	1	0,1	Terminado		VOL-INS-NBLAN	
VOL_INS_06	Ingresar el dominio de idiomas	CM	1	0,1	Terminado		VOL-INS-NBLAN	
VOL_INS_07	Ingresar el campo de preferencia de voluntariado	CM	1	0,1	Terminado		VOL-INS-NBLAN	
VOL_INS_08	Establecer la contraseña (2 campos)	CM	1	0,3	Terminado	A1-D1-01/A2-D2-01 A3-D3-01/A3-D3-02 A3-D3-03	VOL-INS-NBLAN VOL-INS-PASS	SEG-INT-01
VOL_INS_09	Ingresar Teléfono	CM	1	0,1	Terminado	A1-D1-01	VOL-INS-NBLAN	

Firmas: (GM) (DUEÑO DE PRODUCTO)

Metodología ágil para desarrollo de software web seguro

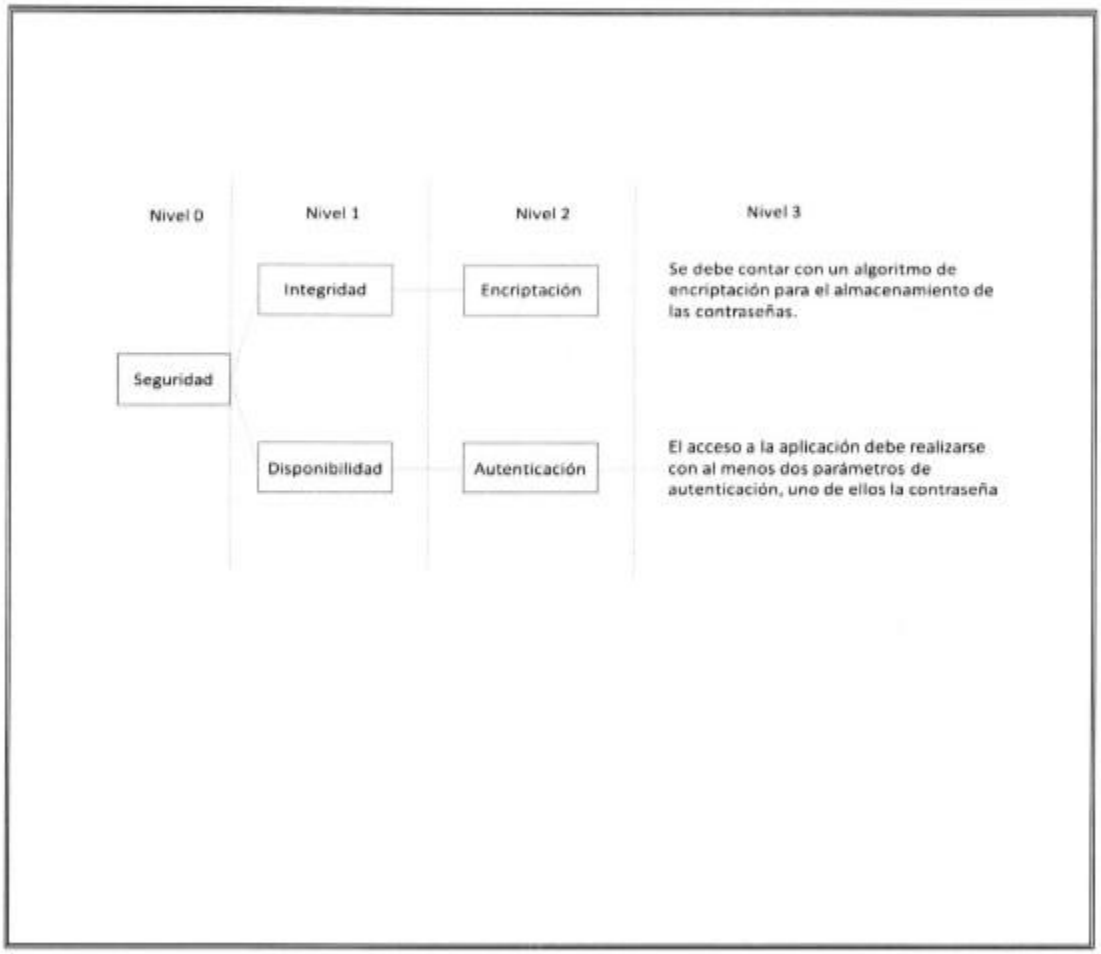
10.2.3. Árbol de Utilidad

Scowp

Árbol de Utilidad

Espacio Reservado para Logo Institucional

Fecha: 17/10/2018
Id. Producto: VOL-INS
Descripción del Producto Administración de base de voluntarios
Id. Utilidad : SEG **Nombre:** Seguridad



Guía Metodológico: Augusto De Arco Firma:

Resp. Equipo Pruebas: Darwin Plaza A. Firma:

Dueño del producto: ESTUDIANTES MIEMBROS Firma:

10.2.4. Escenarios de calidad

Escenarios de calidad

Espacio Reservado para Logo Institucional

Id. Producto VOL-INS
Descripción Producto Administración de Base de Voluntarios

Id. Atributo de utilidad SEG

Id. Escenario de calidad	SEG-INT-01
Descripción escenario: Si el usuario intenta acceder	
Generador del estímulo	Administrador
Estímulo	Revisión de la función de encripción de contraseñas
Entorno	Sistema
Componente	Módulo de registro de contraseñas
Respuesta	Aprobación de la función de encripción
Medida de la respuesta	Inmediata

Id. Escenario de calidad	SEG-DIS-01
Descripción escenario: Todo acceso al sistema, sin distinguir tipo de usuario, debe cumplir con la verificación de al menos dos parámetros incluyendo la contraseña	
Generador del estímulo	Administrador
Estímulo	Revisión del proceso de autenticación
Entorno	Sistema
Componente	Módulo de acceso de usuarios al sistema
Respuesta	Aprobación/Objeción del proceso
Medida de la respuesta	Inmediata

Id. Escenario de calidad	SEG-DIS-02
Descripción escenario: El proceso de acceso al sistema debe suspenderse cuando el usuario ingrese credenciales erradas	
Generador del estímulo	Usuario
Estímulo	Acceso al sistema con credenciales erradas
Entorno	Sistema
Componente	Módulo de autenticación de usuarios
Respuesta	No permitir la entrada al sistema por parte del usuario solicitante y borrar los campos de entrada de credenciales
Medida de la respuesta	Inmediata

Guía Metodológico: Augusto De Arco Firma:

Resp. Equipo Pruebas: Ramiro Flora Firma:

Dueño del producto: ATENOGENES Firma:

10.2.5. Casos de pruebas funcionales

Casos de pruebas funcionales

Espacio Reservado para Logo Institucional

Id. Pendiente VOL-INS
Sprint 1

Id. P. Funcional: VOL-INS-IDEN

Descripción de la actividad de prueba
Ingresar una nacionalidad e identificación existentes en el sistema.

Condiciones previas del sistema
El sistema debe contener la dupla Nacionalidad/identificación igual a la que se está ingresando.

Respuesta esperada del sistema
El sistema debe mostrar un mensaje que el voluntario ya existe y el proceso debe detenerse, limpiar los campos ingresados y quedar listo para que se ingresen otros datos.

Id. P. Funcional: VOL-INS-EMAIL
Descripción de la actividad de prueba
Ingreso de correo electrónico

Condiciones previas del sistema
El sistema debe contener un registro de voluntario con un correo electrónico igual al que se intenta ingresar.

Respuesta esperada del sistema
Debe enviar un mensaje de "Correo electrónico existente en el sistema" y permitir cambiarlo dejando el cursor en el campo de ingreso.

Id. P. Funcional: VOL-INS-PASS
Descripción de la actividad de prueba
Ingresar una contraseña que no cumpla con los requisitos de contar con al menos un número, una mayúscula y un carácter especial y que no cuente con los 8 caracteres mínimos de longitud.

Condiciones previas del sistema
Contar con la función evaluadora de contraseñas

Respuesta esperada del sistema
No permitir el establecimiento de la contraseña mediante un mensaje al usuario y regresar el cursor al primer campo de ingreso de la contraseña

Firma Guia metodológico:

Firma Resp. Pr. Func.

Metodología ágil para desarrollo de software web seguro

Scowp

Casos de pruebas funcionales

Espacio Reservado para Logo Institucional

Id. Pendiente VOL-INS
Sprint 1

Id. P. Funcional: VOL-INS-NBLAN

Descripción de la actividad de prueba
Dejar el campo en blanco e intentar Grabar la información.

Condiciones previas del sistema
Tener habilitado el botón de Grabar para almacenar la información ingresada en el registro de voluntarios.

Respuesta esperada del sistema
El sistema debe indicar a traves de un mensaje que el campo debe ser diligenciado obligatoriamente. Debe señalar el primer campo en blanco encontrado dentro de la secuencia de diligenciamiento.

Id. P. Funcional: VOL-INS-VPAS

Descripción de la actividad de prueba
Ingresar dos contraseñas diferentes en los campos establecidos para establecer la contraseña.

Condiciones previas del sistema
Contar con una función de comparación de los dos campos ingresados

Respuesta esperada del sistema
El sistema debe indicar a traves de un mensaje que los campos no coinciden y que deben ser iguales. Dejar el cursor en el primer campo de contraseña a ingresar.

Id. P. Funcional: VOL-INS-GRAB
Descripción de la actividad de prueba
Grabar la información del voluntario en la base de datos

Condiciones previas del sistema
Haber validado el cumplimiento de todos las necesidades funcionales y no funcionales establecidas para cada actividad del pendiente.

Respuesta esperada del sistema
Registrar en la Base de Datos la información del nuevo voluntario siguiendo las recomendaciones OWASP asociadas a los riesgos.

Firma Guia metodológico:

Firma Resp. Pr. Func.

Scowp

Casos de pruebas funcionales

Espacio Reservado para Logo Institucional

Id. Pendiente VOL-INS
Sprint 1

Id. P. Funcional: VOL-INS-CANC

Descripción de la actividad de prueba

Cancelar la inscripción del voluntario.

Condiciones previas del sistema

Estar en el proceso de inscripción del voluntario.

Respuesta esperada del sistema

Limpiar todos los campos diligenciados y dejar el cursor en el campo de Nacionalidad.

Id. P. Funcional:

Descripción de la actividad de prueba

Condiciones previas del sistema

Respuesta esperada del sistema

Id. P. Funcional:

Descripción de la actividad de prueba

Condiciones previas del sistema

Respuesta esperada del sistema

Firma Guia metodológico:

Firma Resp. Pr. Func.

10.2.6. Casos de pruebas OWASP-RVD

Casos de pruebas OWASP-RVD

Espacio Reservado para Logo Institucional

Año de Matriz OWASP-RVD: 2018
Id. Riesgo OWASP: A1

Id. Caso de prueba: A1-D2-01

Descripción de la actividad de prueba
Ingresar una sentencia que genere comparación de datos o comando SQL

Condiciones previas del sistema
Contar con una función de análisis de cadena de carateres digitada que detecte las sentencias ocultas en el ingreso por parte del usuario

Respuesta esperada del sistema
Rechazo de la información ingresada a través de un mensaje al usuario, limpiar el campo y permitir nuevamente la digitación de caracteres.

Id. Caso de prueba:
Descripción de la actividad de prueba

Condiciones previas del sistema

Respuesta esperada del sistema

Id. Caso de prueba:
Descripción de la actividad de prueba

Condiciones previas del sistema

Respuesta esperada del sistema

Firma Guia metodológico:

Firma Resp. Equipo Pruebas

Metodología ágil para desarrollo de software web seguro

Scowp

Casos de pruebas OWASP-RVD

Espacio Reservado para Logo Institucional

Año de Matriz OWASP-RVD: 2018
Id. Riesgo OWASP: A2

Id. Caso de prueba: A2-D2-01

Descripción de la actividad de prueba

Ingresar una contraseña que esté conformada por uno de los siguientes parámetros: cadena de identificación del correo electrónico (antes del @), cadena del nombre, cadena del apellido, secuencia numérica consecutiva.

Condiciones previas del sistema

Contar con una función de análisis de cadena para evaluar el ingreso de la posible contraseña.

Respuesta esperada del sistema

Envío de mensaje al usuario que la contraseña incumple una de las condiciones y dejar claridad de cuál de ellas se está violando. Volver a colocar el cursor en el primer campo de petición de la contraseña.

Id. Caso de prueba: A2-D3-01

Descripción de la actividad de prueba

Ingresar contraseñas no contengan Mayúsculas en su cadena

Condiciones previas del sistema

Contar con una función de análisis de cadena para evaluar el ingreso de la posible contraseña.

Respuesta esperada del sistema

Envío de mensaje al usuario que la contraseña incumple una de las condiciones y dejar claridad de cuál de ellas se está violando. Volver a colocar el cursor en el primer campo de petición de la contraseña.

Id. Caso de prueba: A2-D3-02

Descripción de la actividad de prueba

Ingresar contraseñas no contengan números en su cadena

Condiciones previas del sistema

Contar con una función de análisis de cadena para evaluar el ingreso de la posible contraseña.

Respuesta esperada del sistema

Envío de mensaje al usuario que la contraseña incumple una de las condiciones y dejar claridad de cuál de ellas se está violando. Volver a colocar el cursor en el primer campo de petición de la contraseña.

Firma Guia metodológico: _____

Firma Resp. Equipo Pruebas _____

Casos de pruebas OWASP-RVD

Espacio Reservado para Logo Institucional

Año de Matriz OWASP-RVD: 2018
Id. Riesgo OWASP: A2

Id. Caso de prueba: A2-D3-03

Descripción de la actividad de prueba
Ingresar una contraseña que no contenga caracteres especiales.

Condiciones previas del sistema
Contar con una función de análisis de cadena para evaluar el ingreso de la posible contraseña.

Respuesta esperada del sistema
Envío de mensaje al usuario que la contraseña incumple una de las condiciones y dejar claridad de cuál de ellas se está violando. Volver a colocar el cursor en el primer campo de petición de la contraseña.

Id. Caso de prueba:
Descripción de la actividad de prueba

Condiciones previas del sistema

Respuesta esperada del sistema

Id. Caso de prueba:
Descripción de la actividad de prueba

Condiciones previas del sistema

Respuesta esperada del sistema

Firma Guia metodológico:

Firma Resp. Equipo Pruebas

Casos de pruebas OWASP-RVD

Espacio Reservado para Logo Institucional

Año de Matriz OWASP-RVD: 2018
Id. Riesgo OWASP: A3

Id. Caso de prueba: A3-D3-01

Descripción de la actividad de prueba

Grabar un segmento de campos que han sido evaluados y cumplen con las condiciones del sistema

Condiciones previas del sistema

Que el sistema cuente con funciones de encriptación de datos en la transmisión durante la grabación.

Respuesta esperada del sistema

Que los datos sean grabados correctamente usando un protocolo de transmisión encriptado.

Id. Caso de prueba: A3-D4-01

Descripción de la actividad de prueba

Ingresar una contraseña correcta

Condiciones previas del sistema

Que el sistema acepte la contraseña y proceda a grabar mediante el uso de funciones y algoritmos de encriptación.

Respuesta esperada del sistema

El sistema debe grabar la contraseña y almacenarla encriptada.

Id. Caso de prueba:

Descripción de la actividad de prueba

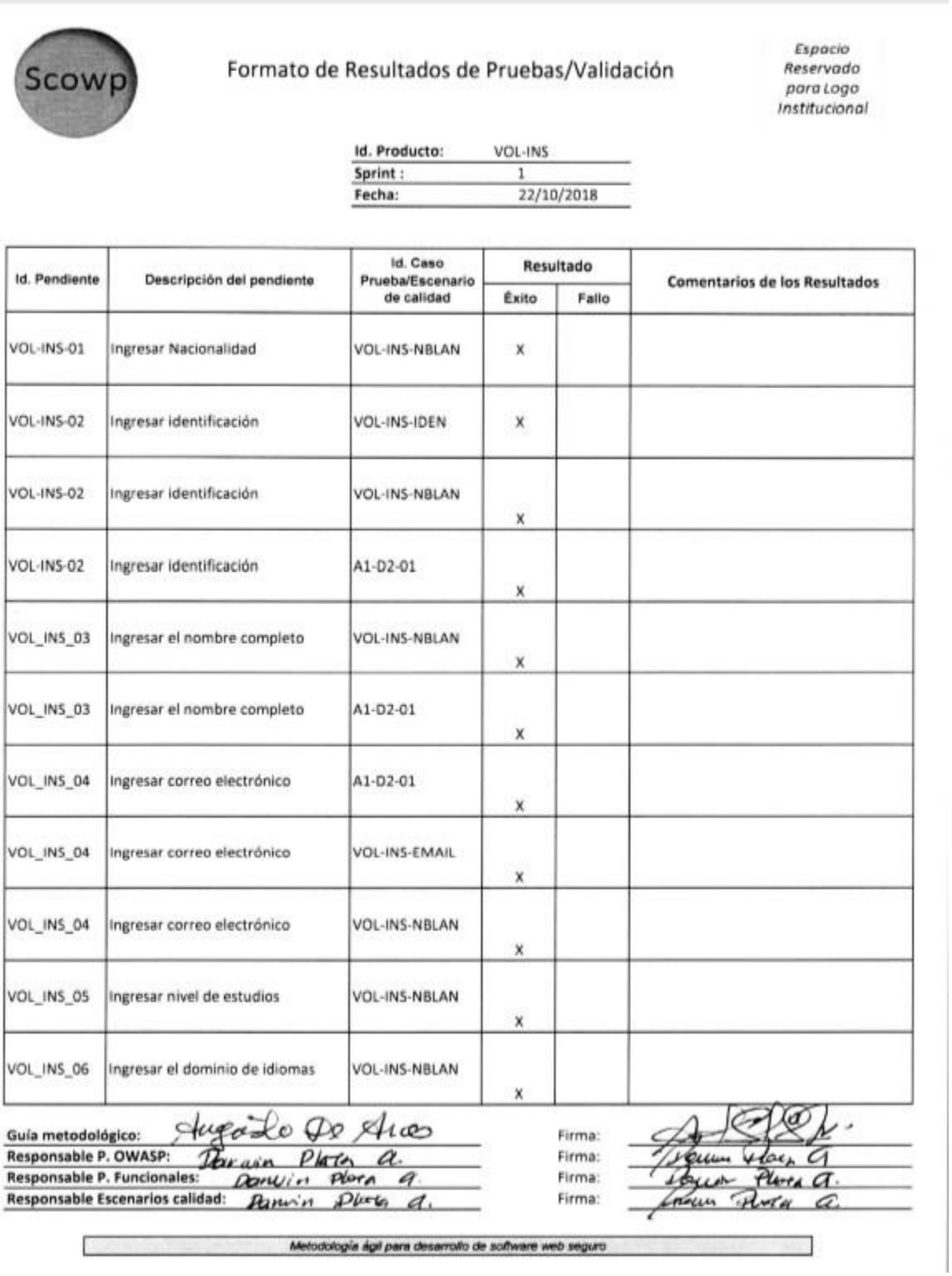
Condiciones previas del sistema

Respuesta esperada del sistema

Firma Guia metodológico:

Firma Resp. Equipo Pruebas

10.2.7. .Formato de resultado de pruebas/validación



Scowp

Formato de Resultados de Pruebas/Validación

Espacio Reservado para Logo Institucional

Id. Producto:	VOL-INS
Sprint :	1
Fecha:	22/10/2018

Id. Pendiente	Descripción del pendiente	Id. Caso Prueba/Escenario de calidad	Resultado		Comentarios de los Resultados
			Éxito	Fallo	
VOL-INS-01	Ingresar Nacionalidad	VOL-INS-NBLAN	X		
VOL-INS-02	Ingresar identificación	VOL-INS-IDEN	X		
VOL-INS-02	Ingresar identificación	VOL-INS-NBLAN	X		
VOL-INS-02	Ingresar identificación	A1-D2-01	X		
VOL_INS_03	Ingresar el nombre completo	VOL-INS-NBLAN	X		
VOL_INS_03	Ingresar el nombre completo	A1-D2-01	X		
VOL_INS_04	Ingresar correo electrónico	A1-D2-01	X		
VOL_INS_04	Ingresar correo electrónico	VOL-INS-EMAIL	X		
VOL_INS_04	Ingresar correo electrónico	VOL-INS-NBLAN	X		
VOL_INS_05	Ingresar nivel de estudios	VOL-INS-NBLAN	X		
VOL_INS_06	Ingresar el dominio de idiomas	VOL-INS-NBLAN	X		

Guía metodológico: Augusto De Arco — Firma:

Responsable P. OWASP: Ronwin Flora A. — Firma:

Responsable P. Funcionales: Ronwin Flora A. — Firma:

Responsable Escenarios calidad: Ronwin Flora A. — Firma:

Metodología ágil para desarrollo de software web seguro

Formato de Resultados de Pruebas/Validación

Espacio Reservado para Logo Institucional

Id. Producto:	VOL-INS
Sprint :	1
Fecha:	22/10/2018

Id. Pendiente	Descripción del pendiente	Id. Caso Prueba/Escenario de calidad	Resultado		Comentarios de los Resultados
			Éxito	Fallo	
VOL-INS-07	Ingresar el campo de preferencia de voluntariado	VOL-INS-NBLAN	X		
VOL-INS-08	Ingresar contraseña (2 campos)	VOL-INS-NBLAN	X		
VOL-INS-08	Ingresar contraseña (2 campos)	VOL-INS-PASS	X		
VOL-INS-08	Ingresar contraseña (2 campos)	SEG-INT-01	X		
VOL-INS-08	Ingresar contraseña (2 campos)	A1-D2-01	X		
VOL-INS-08	Ingresar contraseña (2 campos)	A2-D3-01	X		
VOL-INS-08	Ingresar contraseña (2 campos)	A2-D3-02	X		
VOL-INS-08	Ingresar contraseña (2 campos)	A2-D3-03	X		
VOL-INS-08	Ingresar contraseña (2 campos)	A3-D4-01	X		
VOL-INS-09	Ingresar número de teléfono	A1-D2-01	X		
VOL-INS-09	Ingresar número de teléfono	VOL-INS-NBLAN	X		
VOL-INS-10	Ingresar dirección de residencia	A1-D2-01	X		
VOL-INS-10	Ingresar dirección de residencia	VOL-INS-NBLAN	X		
VOL-INS-11	Ingresar municipio de residencia	A1-D2-01	X		
VOL-INS-11	Ingresar municipio de residencia	VOL-INS-NBLAN	X		

Guía metodológico:	Arquelio De Arco.	Firma:
Responsable P. OWASP:	Darwin Plata A.	Firma:
Responsable P. Funcionales:	Darwin Plata A.	Firma:
Responsable Escenarios calidad:		Firma:

Formato de Resultados de Pruebas/Validación

Espacio Reservado para Logo Institucional

Id. Producto:	VOL-INS
Sprint :	1
Fecha:	22/10/2018

Id. Pendiente	Descripción del pendiente	Id. Caso Prueba/Escenario de calidad	Resultado		Comentarios de los Resultados
			Éxito	Fallo	
VOL-INS-12	Grabar información	VOL-INS-NBLAN	X		
VOL-INS-12	Grabar información	A3-D3-01	X		
VOL-INS-12	Grabar información	A3-D4-01	X		
VOL-INS-12	Grabar información	SEG-INT-01	X		
VOL-INS-13	Cancelar información	VOL-INS-CANC	X		

Guía metodológico: Augusto De Arco — Firma:

Responsable P. OWASP: Dorwin Plata A. — Firma:

Responsable P. Funcionales: Dorwin Plata A. — Firma:

Responsable Escenarios calidad: Dorwin Plata A. — Firma:

10.2.8. Aceptación de producto “Terminado”

Aceptación de producto "Terminado"

Espacio Reservado para Logo Institucional

Fecha: 23/10/2018
Año/Sprint 2018 / 01

Dentro de la revisión del Sprint a quienes asistieron:

César Orlando Mattos Badillo
Atenógenes Miranda Carrillo
Darwin Plata
Augusto De Arco Chiquillo

se determinó que los siguientes productos fueron aprobados y se consideran "Terminados" y listos para ser Incrementos del sistema actual:

Id. Producto	Descripción del producto
VOL_INS	Inscripción de Voluntarios

Guia Metodológico: Augusto De Arco Firma:

Dueño del producto: ATENOGENES M Firma: