



DESARROLLO DE INTERFAZ GRÁFICA Y DESARROLLO CONTINUO
PARA EL PRODUCTO EN DESARROLLO WEB SECURITY MANAGER
(WSM) DE LA EMPRESA
ALERT LOGIC INC COLOMBIA
Modalidad Pasantía

OLIVER ZULUAGA MONSALVE
CÓDIGO: 1527391
oliver.zuluaga@correounivalle.edu.co

Director en la Empresa
JOHN GOMEZ CARVAJAL
Manager de Security Operations Center Tools
jgomez@alertlogic.com

Director EISC
JOHN ALEXANDER SANABRIA ORDOÑEZ
Ph.D. en Computer Science and Engineering
john.sanabria@correounivalle.edu.co

Facultad de Ingeniería
Escuela de Ingeniería de Sistemas y Computación
Programa Académico de Ingeniería de Sistemas
Tuluá, Junio de 2016

Nota de aceptación:

Firma del Coordinador

Firma del Jurado 1

Firma del Jurado 2

Cali- valle del cauca, noviembre 2017

Agradecimientos

Quiero agradecer al docente John Sanabria por su acompañamiento en este proceso de constante aprendizaje, siendo un guiador y director de mi proyecto.

A mi manager John Gomez por darme la oportunidad de incrementar mis conocimientos en la empresa Alert Logic inc Colombia, por enseñarme que el trabajo en equipo, la responsabilidad, la comunicación y el cumplimiento de entrega en cada actividad son lo más importante para el desarrollo de cada día. Esto ayuda a que el desarrollo de mis capacidades sean más eficaces y sobre todo me hace crecer tanto en lo personal como en lo profesional.

A mis compañeros John Leyton Diaz, Danny Salazar, Oscar Plaza y Óscar Restrepo por compartir su conocimiento, por ayudarme, por ser guiadores y mostrarme el valor de trabajar en equipo durante mi pasantía.

A la universidad del Valle, junto a sus destacados maestros por brindarme la oportunidad de formarme como profesional, dándome bases sólidas, una formación integral haciéndome crecer y madurar.

CONTENIDO

1 Introducción	5
1.1 Descripción general	5
1.2 Descripción de la empresa	5
1.3 Planteamiento del problema	6
1.4 Objetivos de la pasantía	7
1.4.1 Objetivo general	7
1.4.2 Objetivos específicos	7
2 Marco de referencia	8
2.1 Modelo de negocios	8
2.1.1 Cloud Computing	8
2.1.2 Software as a Service	8
2.1.1.1 Modelo de negocios de SaaS	9
2.1.1.2 Factores claves de SaaS	9
2.1.2.3 Estructura operativa de SaaS	10
2.2 Herramientas de desarrollo web	10
2.2.1 AngularJS	10
2.2.2 GIT	11
2.2.3 GitHub	12
2.2.4 Chef	12
2.2.5 Ruby	12
2.2.6 Jenkins	12
2.2.7 Atom	13
2.2.8 Typescript	13
2.2.9 Unit tests	13
2.2.10 Lint	13
2.2.11 NPM	13
2.2.12 Jasmine (JavaScript testing framework)	13
2.2.13 Karma	14
2.2.14 Web Application Firewall (WAF)	14
2.2.15 Proxy	14
2.3 Metodología de desarrollo Scrum	15
2.3.1 Product Backlog	16
2.3.2 Sprint Backlog	16
2.3.3 Incrementos	17
2.3.4 Sprint Retrospective	17

2.3.5 Backlog Grooming	17
3. Desarrollo del proyecto	18
3.1 Conceptos relevantes alrededor del proyecto	18
3.2 Desarrollo objetivo específico I	20
3.3 Desarrollo objetivo específico II	21
3.4 Desarrollo objetivo específico III	25
3.5 Desarrollo objetivo específico IV	26
4. Conclusiones	28
5. trabajos futuros	29
6. Referencias	30

1. Introducción

1.1 Descripción general

El presente trabajo de grado en modalidad de pasantía tiene como objetivo principal presentar un informe detallado de las actividades realizadas dentro de la empresa Alert Logic Inc. Esta, es una empresa de desarrollo de software enfocada a prestar servicios de seguridad en la nube utilizando el modelo *Security as a Service*, la cual, tuvo la iniciativa de crear un programa de pasantía en acuerdo con la Universidad del Valle para adquisición de nuevos talentos dentro de su personal dando así origen al programa de pasantía descrito en el presente documento. Este, consiste en la capacitación de estudiantes del programa de Ingeniería en Sistemas de la Universidad del Valle en las tecnologías, lenguajes de programación, frameworks y forma de trabajo de la empresa durante un periodo de seis meses.

Al inicio de la pasantía se planearon unas actividades las cuales consistieron en la contextualización del proyecto respecto a las capacitaciones requeridas para participar en el desarrollo del proyecto, modelo de trabajo, herramientas de desarrollo a usadas, lenguajes de programación, la implementación de pruebas unitarias sobre el producto a desarrollar, el despliegue a producción de nuevos productos. Adicionalmente por fuera de estas actividades se realizó la gestión de las historias de usuario.

Todas las actividades mencionadas se limitaron al desarrollo del producto de la empresa y fueron supervisadas por el equipo de Soc Tools de la empresa Alert Logic a cargo del Manager John Gomez Carvajal quien se encargó de verificar el cumplimiento de las tareas y objetivos propuestos.

1.2 Descripción de la empresa

Alert Logic tiene más de una década de experiencia, perfeccionando y redefiniendo soluciones seguras, flexibles, diseñadas para trabajar con proveedores de servicios en modalidad de “hosting” o en la nube “cloud”. Alert Logic entrega una solución completa que, si bien vive en la nube, tiene sus raíces en la experiencia real.

Ya sea que su empresa esté en proceso de migración de su infraestructura a la nube aprovechando los servicios de hosting gestionado, o aún siga confiando en sus propios centros de datos “datacenters”, Alert Logic está en la capacidad de ofrecer soluciones de detección de intrusiones, análisis de vulnerabilidad y gestión de registros que se complementan con el servicio de monitoreo y vigilancia 24 x 7 y los servicios orientación brindada por expertos de nuestro centro de gestión de seguridad [1].

Alert Logic, el líder en soluciones de seguridad y cumplimiento normativo para la nube, ofrece soluciones de Seguridad como Servicio (SaaS, Security-as-a-Service) especialmente indicadas para ser implementadas en infraestructuras con instalaciones dedicadas o datacenters, en la nube, o en infraestructuras híbridas, entregando a los clientes una visión profunda de seguridad y protección continua, a un costo menor que las soluciones de seguridad tradicionales.

Sus redes, sistemas y aplicaciones Web, o donde resida su infraestructura de IT. Alert Logic está asociado con los líderes de plataformas en la nube y servicios de hosting protegiendo más de 3.000 organizaciones a nivel mundial.

Construida a escala de servicios en la nube, nuestra plataforma patentada tiene la capacidad de almacenar petabytes de información, analiza más de 450 millones de eventos e identifica más de 60.000 incidentes de seguridad cada mes, siendo esto todo gestionado por nuestro centro de Gestión de Seguridad 24 x 7. Alert Logic, fundada en 2002 tiene su sede principal en Houston, Texas, con oficinas en Seattle, Cardiff, Cali y Londres.

1.3 Planteamiento del problema

WSM es uno de los productos principales que provee seguridad a las aplicaciones Web es llamada Web Security Manager WSM. Es básicamente un sistema inteligente que bloquea tráfico malicioso que pasa a través de los proxies de los clientes. Para ello, Alert logic instala dispositivos en los clientes los cuales detectan y reaccionan ante este tipo de tráfico. Estos dispositivos requieren de un proceso de aprendizaje de reglas de bloqueo cada vez que se crea un nuevo Sitio Web, o de mantenimiento y ajustes a estas reglas. En este sentido, la empresa cuenta con un equipo de analistas de seguridad quien se encarga de estas tareas para garantizar que los dispositivos están funcionando adecuadamente. WSM actúa como un *Web Application Firewall* (WAF) [ver sección 2.2.14] [16], existen dos soluciones para WSM:

- Inline: modo *Inline* conocido como Web Security Manager Premier (WSMP)
- OOB: modo *out-of-band* conocido como OOB WAF

Los sitios web y las proxies [ver sección 2.2.15] se utilizan indistintamente. Se trata de la aplicación web configurada por el cliente en los dispositivos. Un sitio web puede estar en uno de tres modos: Proteger, Detectar, Pasar. El equipo de seguridad de WSM busca principalmente sitios web para comprender completamente su carga de trabajo y sintonizar a los clientes en un sitio web. Un cliente está recibiendo el valor completo de WSM al tener todos sus sitios web en modo de protección.

Modo de protección: fase en la que el cliente obtiene el valor total del producto WSM en el que el cliente recibe el mayor valor. En esta fase, el dispositivo WSM bloqueará el tráfico o generará un incidente. El equipo de WSM se esfuerza por obtener todos los sitios web para proteger el modo.

Modo de detección: La fase de sintonización del aparato. El analista de WSM Security mantendrá el sitio web en modo de detección hasta que crean que todos los parámetros han sido contabilizados. El analista suele supervisar los sitios web hasta que no han visto falsos positivos en un sitio web durante una semana o más antes de mover el sitio web para proteger el modo.

Modo de paso: El aparato se mantiene en modo de paso, por lo general a petición especial del cliente. No proporciona ningún valor de seguridad adicional y sólo sirve como un proxy.

Como parte de este esfuerzo por mejorar la eficiencia y la calidad de las aplicaciones internas, las cuales son la base para que los equipos del SOC ejecuten sus flujos de trabajo, la empresa tiene la necesidad de crear una herramienta para el equipo de WSM que les provea con la habilidad de controlar adecuadamente el ajuste de los proxies de los clientes.

1.4 Objetivos en la pasantía

1.4.1 Objetivo General

- ✓ Desarrollar nuevas funcionalidades utilizando las metodologías y tecnologías de desarrollo web, de acuerdo a los requerimientos definidos para el desarrollo de la aplicación Web Security Manager (WSM Tuning tool).

1.4.2. Objetivos Específicos

- ✓ Apropiar metodologías y tecnologías de desarrollo web utilizadas en los productos en Alert Logic.
- ✓ Desarrollo de la interfaz gráfica de usuario utilizando la tecnología AngularJS según el diseño realizado por el equipo de experiencia de usuario.
- ✓ Implementar pruebas unitarias específicas para incrementar la cobertura de código en la interfaz gráfica desarrollada.
- ✓ Desarrollo de artefactos necesarios para el desarrollo continuo.

2. MARCO DE REFERENCIA

A continuación se definen los conceptos que hacen parte de las tecnologías involucradas en el desarrollo de la pasantía las cuales definen del alcance y la aplicabilidad dentro del trabajo realizado. Se profundizará en los conceptos propios del desarrollo de la pasantía como lo son: Modelo de negocio, herramientas de desarrollo y metodología usada.

2.1 Modelo de Negocio

2.1.1 Cloud Computing

Cloud computing se refiere básicamente a la entrega bajo demanda de recursos informáticos y aplicaciones a través de internet con un sistema de precios basado en el consumo realizado. [3]

La importancia de usar los servicios “Cloud” por parte de las empresas radica en que estos proporcionan acceso rápido y flexible a recursos informáticos de bajo costo. Gracias a Cloud Computing, las compañías no necesitan realizar grandes inversiones en la adquisición de equipos ni en la administración de los mismos. En lugar de eso, podrá aprovisionar exactamente el tipo y el tamaño de recursos informáticos que necesite para hacer realidad el propósito de la empresa. [3]

Cloud Computing ofrece un método sencillo de obtener acceso a servidores, almacenamiento, bases de datos y una amplia gama de servicios de aplicaciones a través de internet. [3] Los proveedores de cloud computing, son propietarios y responsables del mantenimiento de los equipos para el funcionamiento de las aplicaciones, mientras el cliente se dedica a usar los recursos por medio de una aplicación web.

2.1.2 Software as a Service

Software as a Service, es una arquitectura de desarrollo en la que se pueden distinguir dos categorías:

- *Servicios orientados al negocio:* estos, se ofrecen en las empresas de todos los tamaños y generalmente son soluciones destinadas a procesos propios del negocio, tales como, finanzas, gestión de cadenas de suministros y relaciones con los clientes.
- *Servicios orientados al consumidor:* estos, se ofrecen al público en general y son servicios orientados al consumidor como por ejemplo, correos electrónicos. Generalmente son gratuitos y soportados por la publicidad.

El SaaS basa su funcionamiento en tres pilares fundamentales que son: el modelo de negocio, arquitectura de las aplicaciones y la estructura operacional. Tal como lo muestra la figura 2.1.

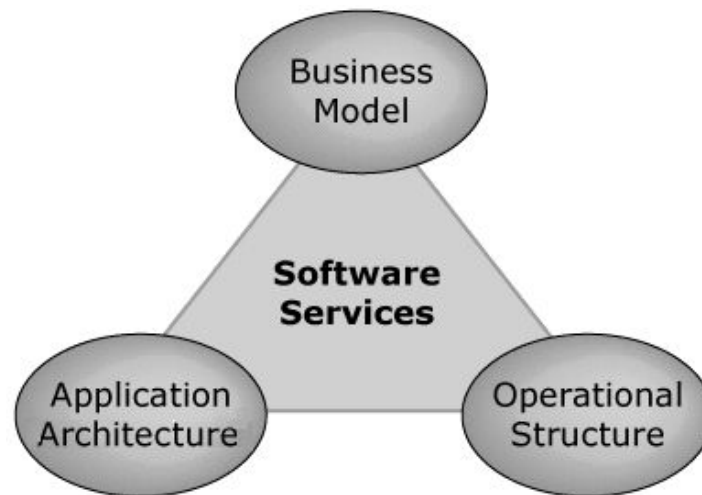


Figura 2.1: Áreas de funcionamiento del SaaS[2]

2.1.2.1 Modelo de negocio de SaaS

Hoy en día, la mayoría de productos de software se sigue vendiendo mediante la modalidad de “software as a product”. Esto consiste en que el cliente compra una licencia para su uso y lo instala en su hardware. Convirtiéndose así en “dueño” del producto.

Con el modelo SaaS, la idea de ser “dueño” del producto se torna un tanto extraña para el cliente ya que en lugar de poseer el producto, se ofrece acceso a este mediante una suscripción. Esto, implica que la aplicación se ejecuta en un servidor externo a la empresa reduciendo así los costos de mantenimiento y soporte.

2.1.2.2 Factores claves de SaaS

Dentro de la arquitectura SaaS hay tres factores claves para asegurar el éxito del producto: La escalabilidad, la eficiencia para atender a múltiples clientes y la configuración.

- *Escalabilidad:* Significa, maximizar la concurrencia y uso de recursos de la aplicación de manera eficiente. Por ejemplo, la distribución de recursos tales como hilos, conexiones de red, almacenamiento en caché y particionamiento de bases de datos.

- *Multitendencia:* Puede ser el cambio más significativo en comparación con la arquitectura de desarrollo tradicional, ya que los usuarios comparten en cierta manera la misma aplicación pero esto no debe afectar el funcionamiento de sus servicios. Para hacer posible esto, se requiere de una arquitectura capaz de maximizar el intercambio de recursos a través de los usuarios pero diferenciando los datos de cada uno. Por tanto no se puede simplemente escribir código personalizado para un solo cliente ya que cualquier cosa que se modifique afecta directamente a los demás clientes.
- *Configuración:* En lugar de la personalización, el modelo SaaS plantea el uso de metadatos para configurar la forma en que aparece la aplicación y se comporta para los usuarios. El reto para implementar esta arquitectura es asegurar que la configuración por parte del cliente sea simple y que no tenga que incurrir en costes de desarrollo adicional.

2.1.2.3 Estructura Operativa de Saas

Otro aspecto importante a tener en cuenta dentro del modelo SaaS, es la estructura operativa de la aplicación, es decir, lo que se necesita para mantener disponible la aplicación.

Proporcionar software como servicio añade una capa adicional en función de ejecutar el plan de negocios de la compañía, esto consisten en:

- Seguimiento a los clientes y uso de recursos.
- Restringir el acceso en ciertos momentos del día.
- Vigilar el acceso al sitio.
- Realizar otras funciones con el fin de garantizar una experiencia perfecta para los clientes.

2.2 Herramientas de Desarrollo

2.2.1 AngularJS

AngularJS es un framework de JavaScript soportado por Google para el desarrollo de aplicaciones web dinámicas bajo el modelo *MVW* (Model View Whatever Works for you). [5] Esta herramienta, contiene un conjunto de librerías útiles para el desarrollo de aplicaciones web del lado del cliente que permiten el intercambio de datos con el servidor de forma que este se libera de cargar completamente la página y solo se limita al envío de datos.

2.2.2 GIT

Git es un servicio de control de versiones distribuido diseñado para la gestión eficiente de flujos de trabajo distribuido no lineales. Git fue diseñado y desarrollado inicialmente por Linus Torvalds en 2005 para el desarrollo del kernel de Linux. La licencia de GIT es libre y existen distribuciones oficiales para los sistemas operativos Mac OS X, Windows, Linux y Solaris. [6] La siguiente figura muestra los principales comandos de GIT (véase figura 2.2).

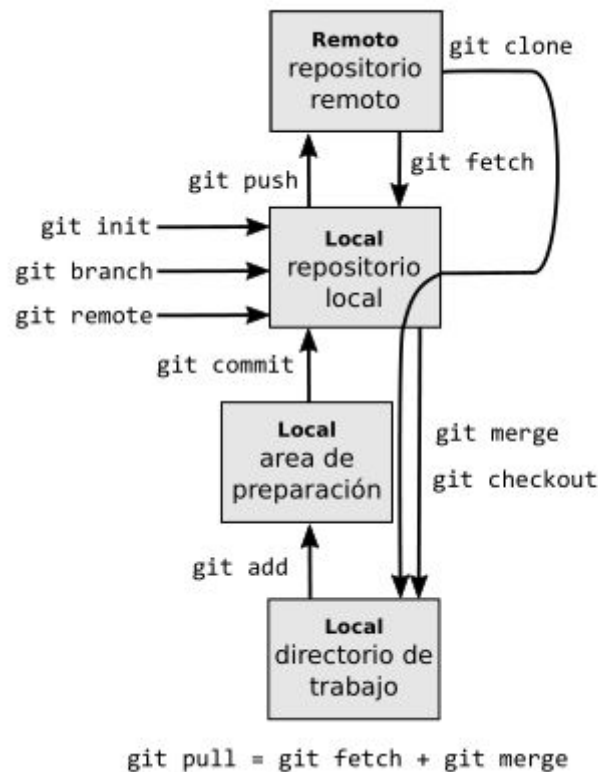


Figura 2.2: Principales comandos de GIT [6]

La experiencia de Linus Torvalds en la gestión de la integración de diferentes aportes en un proyecto distribuido de la magnitud del kernel de Linux determinó las siguientes decisiones de implementación. [6]

- *Versiones no incrementales.* Git almacena cada cambio como una instantánea de todos los archivos del proyecto. Para ser eficiente, si el archivo no ha sido modificado, solo se almacena un enlace al archivo idéntico previamente almacenado.
- *Autenticación criptográfica de la historia.* El identificador de cambio se computa utilizando un algoritmo criptográfico que utiliza como entrada el cambio y la historia completa de cambios. Esto permite que cualquier cambio sea detectado por Git
- *Trabajo fuera de línea.* Por ser un distribuido, cada repositorio Git es un repositorio completo capaz de funcionar sin acceso a la red o al resto de

repositorios.

2.2.3. GitHub

GitHub es un servicio comercial de alojamiento de repositorios Git remotos creado en el año 2008. GitHub proporciona una interfaz Web que permite al usuario registrado crear repositorios vacíos o por clonación de otro repositorio hospedado en GitHub, enviar solicitudes de cambio entre repositorios y gestionar dichas solicitudes. Además, los repositorios en GitHub pueden actuar como repositorios remotos de repositorios locales. Los repositorios que se crean en GitHub son por defecto de acceso público. [7] Sin embargo por medio de una cuenta de pago se pueden alojar repositorios privados. Cada repositorio es propiedad de una cuenta de usuario o de organización. Las modificaciones a un repositorio solo se pueden realizar por usuarios que ha iniciado sesión y que estén autorizados a modificar su contenido.

Además del alojamiento, GitHub proporciona a cada repositorio una wiki, un gestor de tareas, un completo sistemas de gestión de comentarios, un cuadro de control con grafos sociales e, incluso una página web propia. Finalmente, todo este contenido puede ser accedido y manipulado mediante un API web.

2.2.4. CHEF

Chef es una herramienta de gestor de configuraciones escrito en *Ruby* y *Erlang*. *Chef* usa *Ruby* puro para escribir configuraciones de sistemas (recetas), las cuales describen como *Chef* gestiona las aplicaciones y utilidades del servidor; y como estas son configuradas. Estas recetas pueden ser agrupadas en *cookbooks* para un manejo más sencillo y describe una serie de recursos que deberían estar en un estado particular [14].

2.2.5 Ruby

Ruby es un lenguaje de programación interpretado, reflexivo y orientado a objetos, creado por el programador japonés Yukihiro "Matz" Matsumoto [13].

2.2.6 Jenkins

Jenkins es un software open source de integración continua, Jenkins proporciona cientos de complementos para apoyar la construcción, implementación y automatización de cualquier proyecto, puede ser usado como un simple servidor o convertirse en el centro de entrega continua para cualquier proyecto[8].

2.2.7 Atom

Atom es un editor de texto que es moderno, accesible, pero que se puede descifrar hasta el núcleo, una herramienta que puedes personalizar para hacer cualquier cosa, pero también utilizarla de forma productiva sin tocar un archivo de configuración [9].

2.2.8 Typescript

Typescript es un superconjunto de JavaScript con lenguaje de programación libre y código abierto, desarrollado y mantenido por Microsoft. Compila a JavaScript plano, es usado para desarrollar aplicaciones javascript que se pueden ejecutar del lado del cliente o del servidor[10].

2.2.9 Unit Tests

Unit tests es una forma de comprobar el correcto funcionamiento de una unidad de código para verificar que cada parte del código, por separado, está funcionando de la manera correcta, para ello se verifica de acuerdo a una entrada una respuesta [11].

2.2.10 Lint

Lint es una herramienta de programación que se usa para detectar código confuso o incompatible; estas funcionan generalmente realizando un análisis estático del código fuente según unas reglas seleccionadas por el equipo de desarrollo con el objetivo de hacer que el código sea más consistente y evitar errores[12].

2.2.11 NPM

NPM es el manejador de paquetes para Javascript de mayor registro de software en el mundo. Posee paquetes de código reutilizable los cuales pueden ser usados para desarrollar proyectos muchos más grandes o herramientas más eficientes [17].

2.2.12 Jasmine (JavaScript testing framework)

Jasmine es un *framework* para la realización de pruebas de código abierto para JavaScript. Su objetivo es correr en cualquier plataforma habilitada para JavaScript, para no interferir en la aplicación ni en el IDE, y para tener una sintaxis fácil de leer [22].

2.2.13 Karma

Karma se encarga de ejecutar los test de Javascript según se vayan construyendo, de tal forma que ante cualquier fallo el desarrollador se dará cuenta de inmediato. Este se puede configurar con Jasmine como *framework* de *testing*. El objetivo principal de Karma es brindar un entorno de prueba productivo a los desarrolladores. El entorno es uno donde no tienen que configurar montones de configuraciones, sino un lugar donde los desarrolladores pueden simplemente escribir el código y obtener retroalimentación instantánea de sus pruebas. Porque obtener comentarios rápidos es lo que lo hace productivo y creativo [23].

2.2.14 Web Application Firewall (WAF)

Un WAF (Web Application Firewall) es un dispositivo hardware o software que permite proteger los servidores de aplicaciones web de determinados ataques específicos en Internet. Se controlan las transacciones al servidor web de nuestro negocio.

Básicamente nos permite evitar (entre otras) los siguientes ataques:

- **Cross-site scripting** que consiste en la inclusión de código script malicioso en el cliente que consulta el servidor web.
- **SQL injection** que consiste en introducir un código SQL que vulnere la Base de Datos de nuestro servidor.
- **Denial-of-service** que consiste en que el servidor de aplicación sea incapaz de servir peticiones correctas de usuarios.

Hay 2 tipos de WAF: Los que se residen en la red (es decir son un elemento más de la red) y los que se basan en el servidor de aplicaciones (residen en el servidor).

Los WAF son elementos complementarios a las medidas de seguridad que soportan los Firewall clásicos [23].

2.2.15 Proxy

Un proxy es un ordenador intermedio que se usa en la comunicación de otros dos. La información (generalmente en Internet) va directamente entre un ordenador y otro. Mediante un proxy, la información va, primero, al ordenador intermedio (proxy), y éste se lo envía al ordenador de destino, de manera que no existe conexión directa entre

el primero y el último [25].

2.3 Metodología de Desarrollo Scrum

Scrum es una metodología de desarrollo ágil creada para el desarrollo de productos de software y es ideal para empresas donde el entorno de trabajo se caracteriza por tener:

- *Incertidumbre*: sobre esta variable se plantea el objetivo que se quiere alcanzar sin proporcionar un plan detallado del producto.
- *Auto-Organización*: los equipos son capaces de organizarse por sí solos y con autonomía, auto superación y auto-enriquecimiento para afrontar los retos del desarrollo.
- *Control moderado*: se establece un control suficiente para crear un escenario de auto control.
- *Transmisión de Conocimiento*: todos comparten conocimiento y rotan sus funciones a lo largo de la organización.

Scrum tiene como base la idea de creación de ciclos breves para el desarrollo, que comúnmente se llaman *iteraciones* y reciben el nombre de “Sprints” [7].

Para entender el ciclo de desarrollo es necesario conocer las 5 fases que definen el ciclo de desarrollo ágil: [7]

1. *Concepto*: se define de forma general las características del producto y se asigna el equipo encargado de su desarrollo.
2. *Especulación*: en esta fase se hacen disposiciones con la información obtenida y se establecen los límites que marcaran el desarrollo del producto tales como costos y agendas.
Esta fase se repite en cada iteración y consisten en:
 - Desarrollar y revisar los requisitos generales.
 - Mantener la lista de funcionalidades que se esperan
 - Establecer las fechas de las versiones, hitos e iteraciones.
3. *Exploración*: se incrementa el producto en el que se añaden las funcionalidades de la fase de especulación.
4. *Revisión*: el equipo revisa todo lo que se ha construido y se contrasta con el objetivo deseado.
5. *Cierre*: se entregará en la fecha acordada una versión del producto. Al tratarse de una versión, esto no indica la finalización del proyecto, sino que seguirá habiendo cambios, denominados “mantenimiento”, que harán que el producto se acerque al resultado esperado.

Los elementos de la metodología Scrum son:

2.3.1 Product Backlog

Es el inventario en el que se almacenan todas las funcionalidades o requisitos del cliente. Estos, serán los que tendrá el producto al final del ciclo de desarrollo.

Los requisitos serán ordenados en una lista de prioridad por el cliente y el Scrum master, el cual indicará el coste estimado de los requisitos. Es necesario que antes de empezar el primer "Sprint" se definan cuáles van a ser los objetivos del producto y tener la lista de requisitos ya definida, no es necesario que sea muy detallada pero sí lo suficiente para que el equipo pueda trabajar. Finalmente, el Product Backlog irá evolucionando mientras el producto exista en el mercado. [7]

2.3.2 Sprint Backlog

Es la lista de tareas que elabora el equipo durante la planificación de un sprint. Se asigna las tareas a cada persona y el tiempo que queda para terminarlas. De esta manera, el proyecto se descompone en unidades más pequeñas y se puede determinar o ver en qué tareas no se está avanzando. (véase figura 2.3). [7]

Requisito	Tarea	Quien	Estado (No iniciada / en progreso / completada)											
				Dia:	1	2	3	4	5	6	7	8	9	10
				Horas pendientes	1120	1088	1076	1048	1040	1032	1020	1008	992	972
Requisito A	Tarea 1	Joao	Completada		16	8								
Requisito A	Tarea 4	Laura	Completada		4									
Requisito A	Tarea 5	Laura	Completada		4									
Requisito A	Tarea 3	Gabri	Completada		8									
Requisito A	Tarea 2	Laura	Completada		16	8	4							
Requisito A	Tarea 6	Gabri	Completada		8	8	8							
Requisito A	Tarea 7	Joao	Completada		16	16	16	8						
Requisito A	Tarea 8	Laura	Completada		8	8	8							
Requisito A	Tarea 9	Laura	Completada		8	8	8	8	8					
Requisito A	Tarea 10	Laura	Completada		8	8	8	8	8	8	4			
Requisito A	Tarea 11	Joao	Completada		16	16	16	16	16	16	8			
Requisito B	Tarea 12	Gabri	Completada		16	16	16	16	16	16	16	16	8	
Requisito B	Tarea 13	Laura	Completada		16	16	16	16	16	16	16	16	8	
Requisito B	Tarea 14	Joao	En progreso		8	8	8	8	8	8	8	8	8	4
Requisito B	Tarea 15	Gabri	En progreso		8	8	8	8	8	8	8	8	8	8
Requisito B	Tarea 16	Laura	En progreso		8	8	8	8	8	8	8	8	8	8
Requisito C	Tarea 17	Joao	No iniciada		4	4	4	4	4	4	4	4	4	4
Requisito C	Tarea 18	Gabri	No iniciada		8	8	8	8	8	8	8	8	8	8
Requisito C	Tarea 19	Laura	No iniciada		16	16	16	16	16	16	16	16	16	16
Requisito C	Tarea 20	Joao	No iniciada		8	8	8	8	8	8	8	8	8	8

Figura 2.3: ejemplo de sprint backlog[7]

El sprint backlog, se muestra como una lista ordenada por prioridades para el cliente y puede existir independencia entre una tarea u otra. Generalmente, las tareas se suelen gestionar mediante el Scrum Taskboard.

2.3.3 Incrementos

Representa los requisitos que sean completado en una iteración y que son perfectamente operativos. Segundo los resultados obtenidos, el cliente puede realizar cambios cuando estos sean necesarios. [7]

2.3.4 Sprint Retrospective

Es una reunión con el objetivo de mejorar la productividad del equipo y la calidad de los desarrollos en los que se están trabajando, El equipo debe analizar cuál ha sido el desempeño durante la iteración, y cual es el motivo de haber conseguido o no los objetivos con los cuales se comprometió al inicio de la iteración [18].

- Qué cosas han funcionado bien.
- Que cosas se deben mejorar
- Qué cosas han salido mal.
- Que se ha aprendido durante la iteración
- Qué problemas tiene o le están bloqueando el desarrollo de alguna tarea.

Beneficios:

- Incremento de la productividad.
- Aumenta la motivación del equipo al hacer que estos se sientan escuchados y tenidos en cuenta.

2.3.5 Backlog Grooming

Es una reunión en la cual el Product Owner y alguien o todo el equipo revisan los items que estan en el backlog para asegurarse de que estos están organizados por su prioridad haciendo que los que están de primeros están listos para su desarrollo[19].

3. Desarrollo del Proyecto

En la siguiente sección, se va a describir las acciones realizadas en pro de cumplir el objetivo general.

Este capítulo consta primero de el uso que se le dio a la metodología Scrum para desarrollar el proyecto, las secciones 3.1, 3.2, 3.3 y 3.4 presentan los resultados esperados de los objetivos 1, 2, 3 y 4; respectivamente.

Para el desarrollo de las tareas en la pasantía se utilizó la metodología de desarrollo SCRUM Para el desarrollo de los objetivos propuestos en la pasantía fue de importancia la implementación de la metodología Ágil Scrum [ver sección 2.3], la cual permitió coordinar las actividades del equipo en cada sprint y fortalecer el trabajo grupal para los objetivos propuestos. Muchas de las actividades de la metodología fueron implementadas con el objetivo de ayudar con el cumplimiento de los objetivos anteriormente planteados

- **Scrum Daily Meeting:** Esta es una actividad del scrum que se realiza diariamente junto con el equipo de desarrollo en el cual se habla acerca del trabajo que actualmente se viene haciendo. Esta actividad fue de gran importancia porque permitió mantener la comunicación del equipo y permitió el poder tener un espacio para poder comunicar inquietudes al equipo de Soc Tools para resolver problemas o bloqueos que podrían retrasar el cumplimiento de los objetivos.
- **Sprint Planning:** Esta actividad consiste en realizar la planeación de las tareas en las que se van a trabajar durante el sprint y permite al equipo realizar estimaciones sobre el tiempo de cada tarea. Durante la pasantía esta actividad permitió que el equipo conociera sobre las tareas que se iban a ejecutar con el objetivo de conocer cuál es el estado de los proyectos, el ritmo de trabajo de los equipos y realizar una planeación correcta para el cumplimiento de estos.
- **Sprint Demo:** Esta actividad fue de suma importancia para el desarrollo de los objetivos ya que permitió que todo el equipo (Desarrolladores, Managers, y otros directos) conocieran el estado de los proyectos mediante diapositivas y realización de pruebas en vivo para la revisión de estos y dar constancia del cumplimiento de los objetivos. Esta actividad se realiza al finalizar cada sprint.
- **Sprint Retro:** Esta actividad consistió en la evaluación del sprint que finalizó destacando aspectos como: que estuvo mal, que estuvo bien y que se puede mejorar; esto con el objetivo de encontrar falencias en el sprint, oportunidades de mejora y comprometer al equipo a solucionar aquellos aspectos o situaciones que dificultaron el cumplimiento de los objetivos.

Tal como se mencionó, la aplicación del *scrum* fue de vital importancia para el éxito de la pasantía y en el cumplimiento de los objetivos. Se destaca el trabajo del manager John Gómez y de mis compañeros de equipo Oscar Restrepo, Oscar Plaza, John Leyton Diaz, Chance Gibbs y Andre Castro de Zorzo; los cuales brindaron el apoyo en cada etapa de la pasantía y contribuyeron con su gran conocimiento y disposición al cumplimiento de los objetivos.

3.2 Desarrollo Objetivo específico 1

Para la realización del objetivo específico 1 (Apropiar metodologías y tecnologías de desarrollo web utilizadas en los productos en Alert Logic.) la empresa brindó una serie de capacitaciones en: El producto de la compañía *WSM* (Web security manager), Desarrollo de artefactos necesarios para el despliegue de aplicaciones usando *Chef* y *Ruby*; y pruebas unitarias con *Jasmine* y *Karma* (ver sección 2.2.12).

3.2.1 Entrenamiento en el producto de la empresa, *WSM*:

Entre las fechas 10 al 13 de Enero del 2017, el equipo de *Soc Tools* a través de su desarrollador Danny Salazar ofreció una capacitación en el funcionamiento y características de *WSM* facilitando así la adquisición del conocimiento sobre este producto.

Esta capacitación fue fundamental en el desarrollo de la pasantía ya que facilitó conocer las características y cómo funcionaba el producto sobre el cual se iba a trabajar por ejemplo: Cuáles son las 2 soluciones para *WSM*, que significa que un *Appliance* esté en modo *Inline* o *OOB*.

3.2.2 Entrenamiento en pruebas unitarias con *Jasmine* y *Karma*

Entre las fechas 16 al 18 de Enero del 2017, el equipo de *Soc Tools* ofreció una capacitación en el desarrollo de pruebas unitarias usando *Jasmine* y *Karma* por medio de talleres prácticos y ejemplos previamente preparados por el desarrollador de *Soc Tools* Danny Salazar.

Esta capacitación facilitó el conocimiento y habilidades necesarias para realizar el cumplimiento de los objetivos.

3.2.3 Entrenamiento en el desarrollo de artefactos necesarios para el despliegue de aplicaciones:

Entre las fechas 23 al 25 de Enero del 2017, el equipo de Soc Tools a través del desarrollador John Díaz ofreció una capacitación en el desarrollo de artefactos necesarios para el despliegue de aplicaciones y cómo se realizan estos, en comprender cómo funciona la implementación de nuevas funcionalidades dentro de la compañía y aprender a desarrollar recetas usando *Chef*.

Esta capacitación facilitó el conocimiento y habilidades necesarias para realizar despliegues en producción usando las tecnologías acogidas por la empresa.

Para dar constancia del trabajo realizado, la empresa entregó un certificado el cual se encuentra en el anexo ([Anexos Tesis/Anexo Objetivo1/ConstanciaCapacitaciones](#))

3.3 Desarrollo Objetivo específico 2

Este objetivo está relacionado con el desarrollo de la interfaz gráfica de usuario utilizando la tecnología AngularJS según el diseño realizado por el equipo de experiencia de usuario. Fue el principal y más grande objetivo de la pasantía ya que fue aquí donde se realizó el desarrollo de la interfaz de usuario para el proyecto de WSM el cual tuvo como meta mejorar el manejo de los proxies individuales para los clientes por parte de los analistas del SOC. Este se desarrolló usando Angular 2 y consistió en el desarrollo de los siguientes escenarios:

3.3.1 Escenario 1: (visualización de proxies)

Para el desarrollo de este escenario se desarrollaron varios componentes usando Typescript para construir un *dashboard* de *proxies* el cual contiene dos formas posibles para visualizarlos sea en forma de tarjetas (véase figura 3.1) o en forma de lista (véase figura 3.2). Dicha información se trae desde el API la cual en un principio al realizar una solicitud para traer todos los *proxies* desde al API se estaba tomando más tiempo del esperado al traer toda la información completa ya que la empresa actualmente cuenta con más de 10 mil proxies.

Para solucionar este inconveniente se sugirió al equipo de desarrollo que al realizar la solicitud la información se pudiera traer paginada para disminuir los tiempos de carga, por motivos de prioridades del equipo de desarrollo fui encargado de añadir esta característica al API la cual fue desarrollada usando tecnología Node.js.

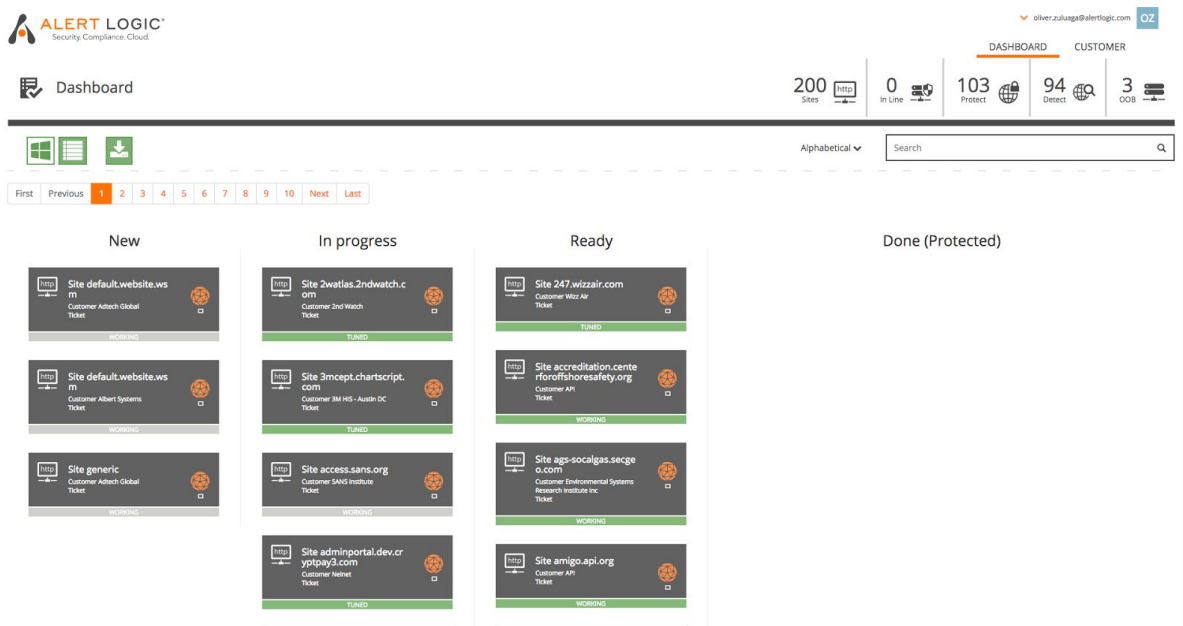


Figura 3.1 Visualizador de Proxies (modo de tarjetas)
Fuente: Elaboración propia

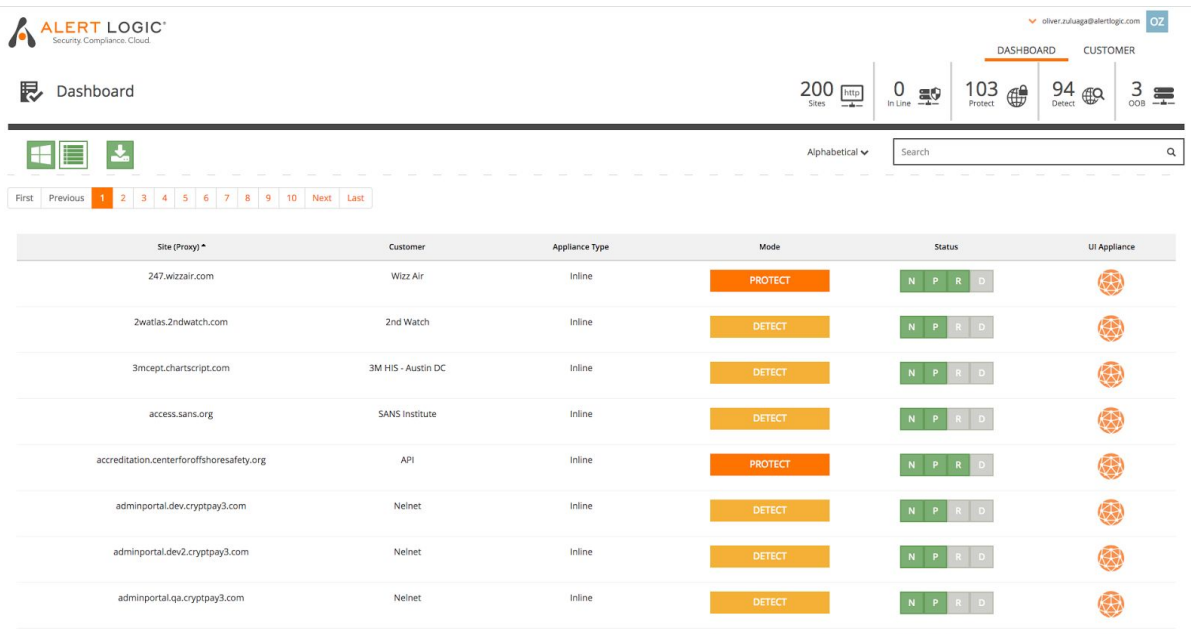


Figura 3.2 Visualizador de Proxies (modo de lista)
Fuente: Elaboración propia

3.3.2 Escenario 2: (Servicios de autenticación)

Para el desarrollo de este escenario se desarrolló un componente para la realización de login (véase figura 3.4) por parte de los analistas el cual se conecta al API para la respectiva autenticación la cual se realiza usando la tecnología LDAP (véase figura 3.3) [20].

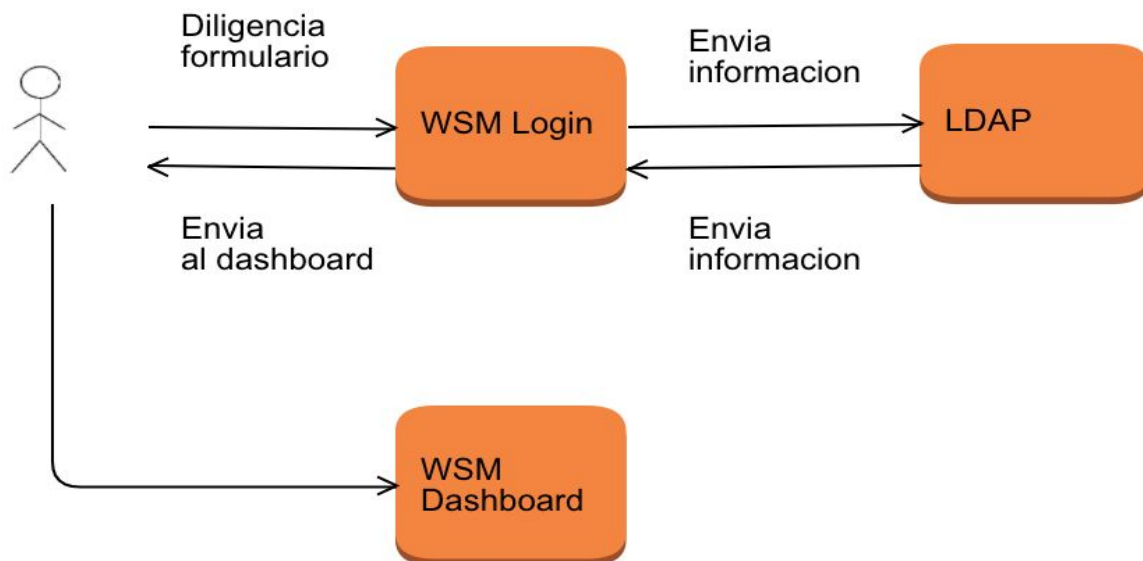


Figura 3.3 Caso de uso explicando los pasos que realizan todos los actores del sistema para realizar una autenticación de usuario

Fuente: Elaboración propia

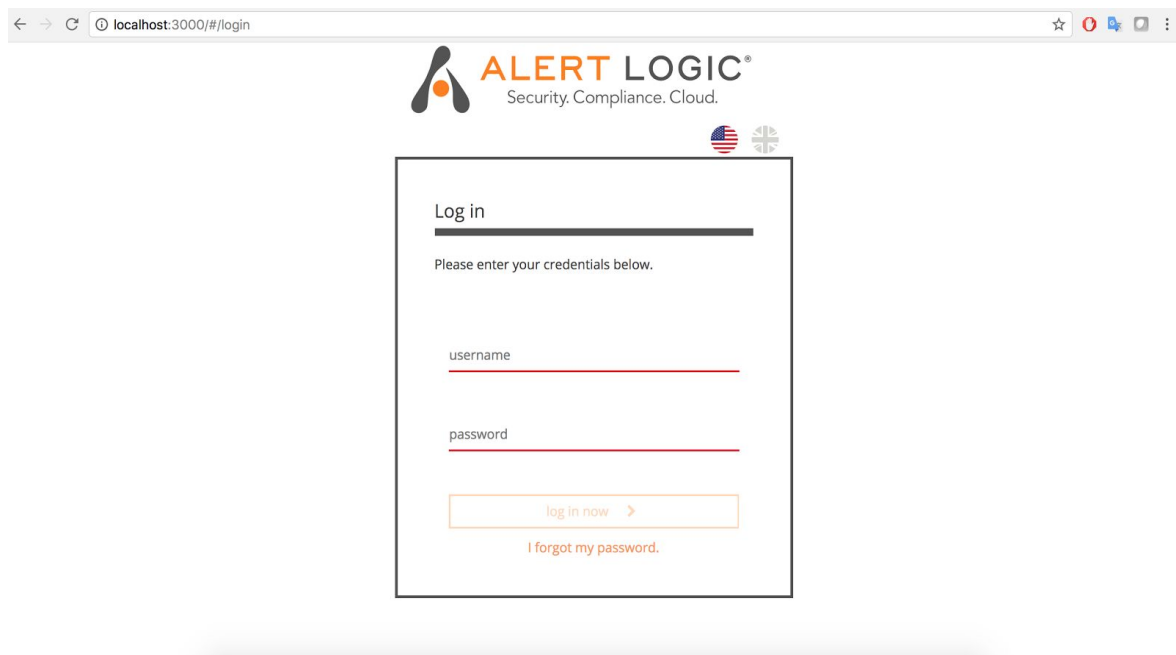


Figura 3.4 Interfaz gráfica del *Login*

Fuente: Elaboración propia

3.3.3 Escenario 3: (Contador de estado de los proxies)

Para el desarrollo de este escenario se desarrolló un componente en el header del dashboard en el cual se contabilizan los proxies según su tipo: (In line, Protect, Detect, Out of Band-OOB) y un totalizador.

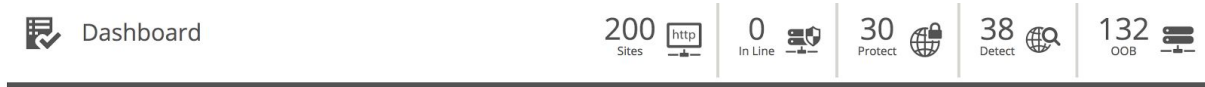


Figura 3.5 Contador de Proxies por estado

Fuente: Elaboración propia

3.3.4 Escenario 4: (Filtros, Ordenamiento y Descarga)

Para el desarrollo de este escenario se desarrolló un componente el cual permite: filtrar los proxies de acuerdo a su nombre (véase figura 3.6) o el nombre del cliente asociado, ordenar los proxies de acuerdo a distintas opciones como lo son:

- Ordenar alfabéticamente por el nombre del Proxy.
- Ordenar alfabéticamente por el nombre del cliente.
- Ordenar alfabéticamente por el tipo del Appliance (*Inline*, *OOB*).
- Ordenar alfabéticamente por el estado (*Protect*, *Detect*, *Pass*).
- Buscar por nombre de proxy o cliente.

The screenshot shows the Alert Logic dashboard with a table of proxies. The table has columns for Site (Proxy), Customer, Appliance Type, Mode, Status, and UI Appliance. The table is filtered by 'Alert Logic' and sorted alphabetically. The table contains 5 rows of data.

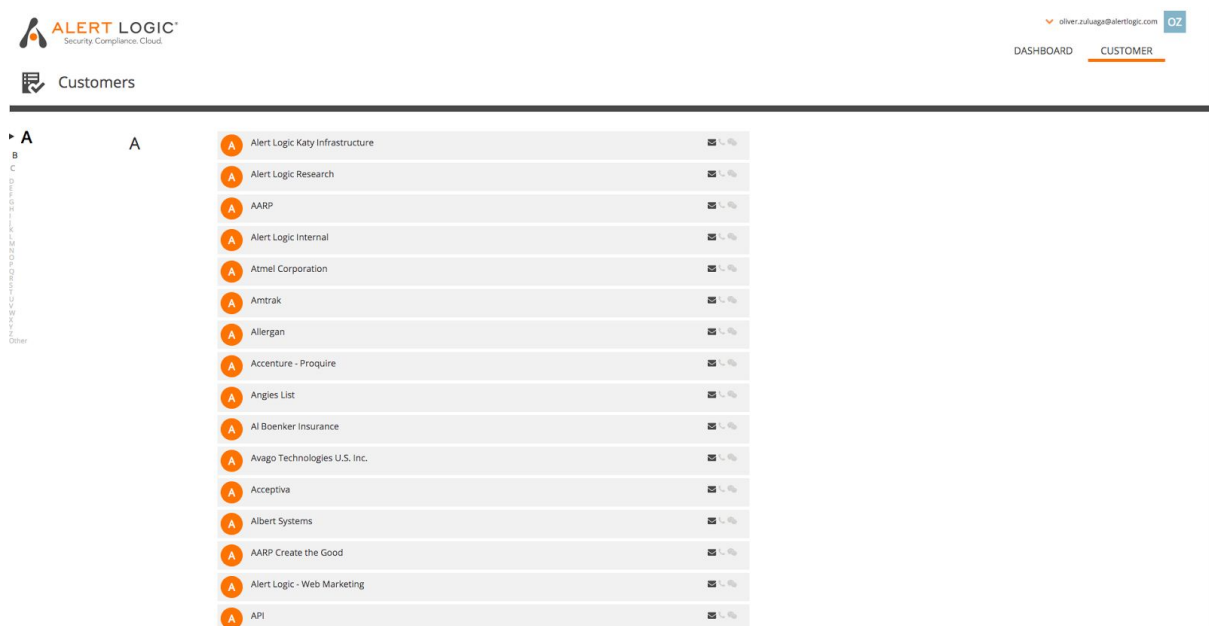
Site (Proxy) *	Customer	Appliance Type	Mode	Status	UI Appliance
console.clouddefender.alertlogic.com	Alert Logic Denver Infrastructure	Inline	DETECT	N P R D	
generic	Alert Logic Denver Infrastructure	Inline	PROTECT	N P R D	
publicapi.alertlogic.net	Alert Logic Denver Infrastructure	Inline	DETECT	N P R D	
updates.profenet.net	Alert Logic Denver Infrastructure	Inline	PROTECT	N P R D	
updates.profenet.net	Alert Logic Denver Infrastructure	Inline	PROTECT	N P R D	

Figura 3.6 Búsqueda de proxies por customer o Proxy

Fuente: Elaboración propia

3.3.5 Escenario 5: (Listado de Clientes)

Para el desarrollo de este escenario se desarrolló un nuevo componente (véase figura 3.7) en el cual se pueden ver los clientes que tienen el servicio WSM y cuál analista es el encargado de un cliente y sus proxies.



	A	
☐	Alert Logic Katy Infrastructure	 
☐	Alert Logic Research	 
☐	AARP	 
☐	Alert Logic Internal	 
☐	Atmel Corporation	 
☐	Amtrak	 
☐	Allergan	 
☐	Accenture - Proquire	 
☐	Angies List	 
☐	AI Boenker Insurance	 
☐	Avago Technologies U.S. Inc.	 
☐	Acceptiva	 
☐	Albert Systems	 
☐	AARP Create the Good	 
☐	Alert Logic - Web Marketing	 
☐	API	 

Figura 3.7 Lista de clientes

Fuente: Elaboración propia

Para más información sobre el cumplimiento del presente objetivo, ver el anexo ([Anexos Tesis/Anexos Objetivos2](#)) en el cual se encuentran más evidencias del desarrollo de los objetivos.

3.4 Desarrollo objetivo específico 3:

El desarrollo del objetivo (Implementar pruebas unitarias específicas para incrementar la cobertura de código en la interfaz gráfica desarrollada) dentro de la pasantía permite que el equipo de Soc Tools pudiera aumentar la cobertura del código la cual es una medida porcentual que mide el grado en que el código fuente de la aplicación ha sido comprobado [21].

El objetivo principal de la implementación de las pruebas fue el de tener una cobertura por encima del 70% ya que esto es uno de los requerimientos solicitados por el comité de cambios en el momento de revisar y aprobar si un cambio va a ser desplegado en producción.

Cabe recalcar que la interfaz gráfica de WSM fue desarrollada usando Angular 2 por lo cual está dividido en componentes ya que esta forma de desarrollo permite reutilizar

código [15] y cada uno de estos tiene asignado procesos independientes dentro del producto y permite al desarrollador trabajar por separado en cada componente. Las herramientas usadas para realizar las pruebas fueron Jasmine que es un framework de pruebas de código abierto para Javascript y Karma que es un lanzador de pruebas que permite automatizar algunas tareas de los frameworks como Jasmine.

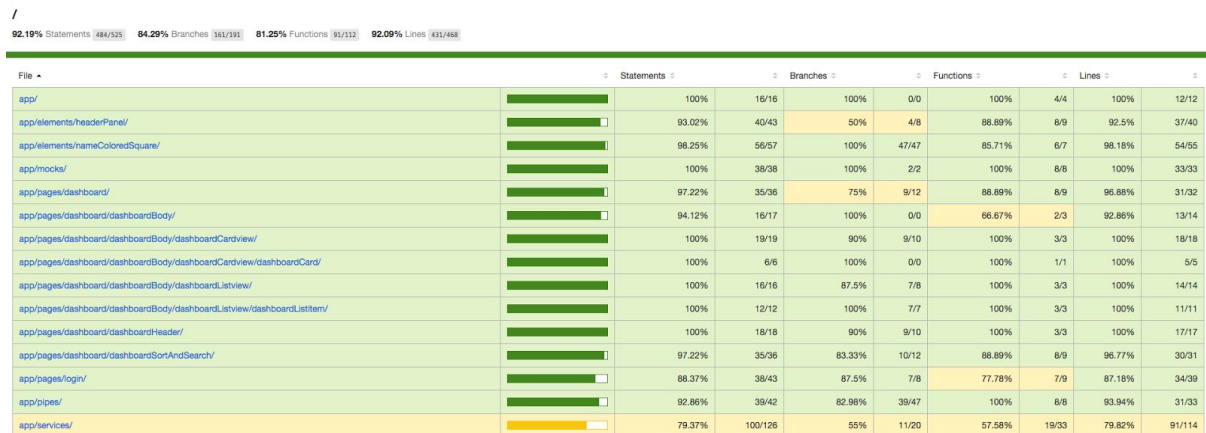


Figura 3.8 Reporte general de pruebas de la aplicación
Fuente: Elaboración propia

Para más información sobre el cumplimiento del presente objetivo, ver el anexo ([Anexos Tesis/Anexos Objetivos3](#)) en el cual se encuentran más evidencias del desarrollo de los objetivos.

3.5 Desarrollo de objetivo específico 4:

El desarrollo del objetivo (Desarrollo de artefactos necesarios para el desarrollo continuo) fue de gran importancia para el desarrollo de la pasantía ya que consistió en el desarrollo de artefactos para realizar el despliegue de la aplicación desarrollada la cual consiste en permitir que el usuario pueda hacer uso de la nueva herramienta o de la actualización realizada de una herramienta ya existente.

Esta tarea permitió adquirir conocimientos sobre el desarrollo continuo de aplicaciones.

- **Definición de nodos:** En este artefacto se especifica todo sobre el nodo que en este caso es una maquina virtual, sistema operativo, información confidencial, recetas de instalacion y configuracion.
- **Receta:** En este artefacto se incluye todo lo necesario para las configuraciones del nodo, se divide en instalacion y configuracion es donde se escoge el software que estará instalado con sus versiones.
- **Secretos:** Es información confidencial la cual no puede ser divulgada y es subida a ambientes de producción para ser cogida desde ahí sin tener conocimiento de ella.

Adicional a eso antes de realizar un despliegue se deben desarrollar 3 artefactos adicionales los cuales son requeridos por el comité de aprobación de cambios para aprobar el despliegue, estos son:

- **Validación de pruebas:** se debe generar un archivo de texto que contenga el registro de las pruebas el cual será revisado por el comité de aprobación de cambios como validación de la implementación de estas.
- **RPN:** hoja de cálculo en la cual se debe llenar un formulario y al final este arrojará un valor de 0 a 100 el cual entre mayor sea determinará mayor riesgo al momento de realizar un despliegue.
- **Scan:** Consiste en realizar un escaneo mediante una herramienta de la empresa a un nodo desplegado en un ambiente de integración para detectar vulnerabilidades.

Después de cumplir estos requerimientos se debe sustentar en una reunión con el comité de aprobación de cambios cuál va a ser el cambio a realizar.

4. Conclusiones

- Como parte del cumplimiento de los objetivos planteados en el presente trabajo es importante mencionar el papel que cumple la implementación de pruebas unitarias en el desarrollo de software a gran escala, ya que durante el desarrollo de estas permitieron identificar bugs que no fueron detectados en el proceso de desarrollo.
- Como aspecto a recalcar es que se mejoró el flujo de trabajo ya que el equipo de WSM tiene una herramienta con la cual pueden ver el estado de sus proxies y tomar acciones en consecuencia.
- Otro aspecto a recalcar es el uso de Github ya que permite llevar un control sobre lo que se ha desarrollado, permitiendo revisiones de código de forma más simple haciendo que el producto final tenga mayor calidad.
- Por último es muy importante recalcar que el desarrollo de este proyecto me permitió conocer y participar en todo el proceso de desarrollo, pruebas y despliegue de una aplicación en un ambiente laboral real e involucrarse en cada uno de los procesos del equipo Soc Tools los cuales me permitieron crecer como profesional del área de la ingeniería de sistemas.

5. Trabajos futuros

Durante el proceso de desarrollo se pudo analizar la posibilidad de desplegar WSM en Amazon Web Services ya que Alert Logic se encuentra en proceso de migrar toda su infraestructura a la nube. En segundo lugar por requerimiento de los analistas también está pendiente evaluar la posibilidad de añadir una funcionalidad adicional que consiste en permitir realizar configuraciones a los proxies desde la interfaz gráfica ya que actualmente solo permite consultar.

6. Referencias

- [1]. Alert Logic – About-us [Online]. Available: <https://www.alertlogic.com/solutions/>
- [2]. Microsoft - Architecture Strategies for Catching the Long Tail. us [Online]. Available: <https://msdn.microsoft.com/en-us/library/aa479069.aspx>
- [3]. Amazon web services - ¿what is cloud computing? [Online]. Available: <https://aws.amazon.com/es/what-is-cloud-computing/>
- [4]. Alert Logic – Solutions [Online]. Available: <https://www.alertlogic.com/assets/cloud-insight/Alert-Logic-Cloud-Insight-How-it-Works-Datasheet.pdf>
- [5]. A. Basalo, M. Alvarez, P. Hurtado and X. Cerdá, “Tutorial de AngularJS”
- [6]. Lopez-Pellicer, F. J., Béjar, R., Latre, M. A., Nogueras-Iso, J., & Zarazaga-Soria, F. J. (2015, July). GitHub como herramienta docente. In *Actas de las XXI Jornadas de la Enseñanza Universitaria de la Informática* (pp. 66-73). Universitat Oberta La Salle.
- [7] Gallego, Manuel Trigas. "Metodología Scrum." *Gestión de Proyectos Informaticos*, 2012 [Online]. Available: <http://openaccess.uoc.edu/webapps/o2/bitstream/10609/17885/1/mtrigasTFC0612memoria.Pdf>.
- [8] <https://jenkins.io/>
- [9] <https://atom.io/>
- [10] <http://www.typescriptlang.org/>
- [11] Unit Testing: [https://msdn.microsoft.com/en-us/library/aa292197\(v=vs.71\).aspx](https://msdn.microsoft.com/en-us/library/aa292197(v=vs.71).aspx)
- [12] <http://eslint.org/docs/user-guide/getting-started>
- [13] <https://www.ruby-lang.org/es/>
- [14] <https://www.chef.io/chef/>
- [15] Desarrollo de Software basado en componentes - <https://msdn.microsoft.com/es-co/library/bb972268.aspx>

- [16] <http://searchsecurity.techtarget.com/definition/Web-application-firewall-WAF>
- [17] <https://www.npmjs.com/>
- [18] <https://proyectosagiles.org/retrospectiva-sprint-retrospective/>
- [19] <https://www.agilealliance.org/glossary/backlog-grooming/>
- [20] <http://web.mit.edu/rhel-doc/4/RH-DOCS/rhel-rg-es-4/ch-ldap.html>
- [21] <http://www.bullseye.com/coverage.html>
- [22] <https://jasmine.github.io/>
- [23] <https://karma-runner.github.io/1.0/index.html>
- [24] [https://www.owasp.org/index.php/Web Application Firewall](https://www.owasp.org/index.php/Web_Application_Firewall)
- [25] <https://www.whatismyip.com/what-is-a-proxy/>