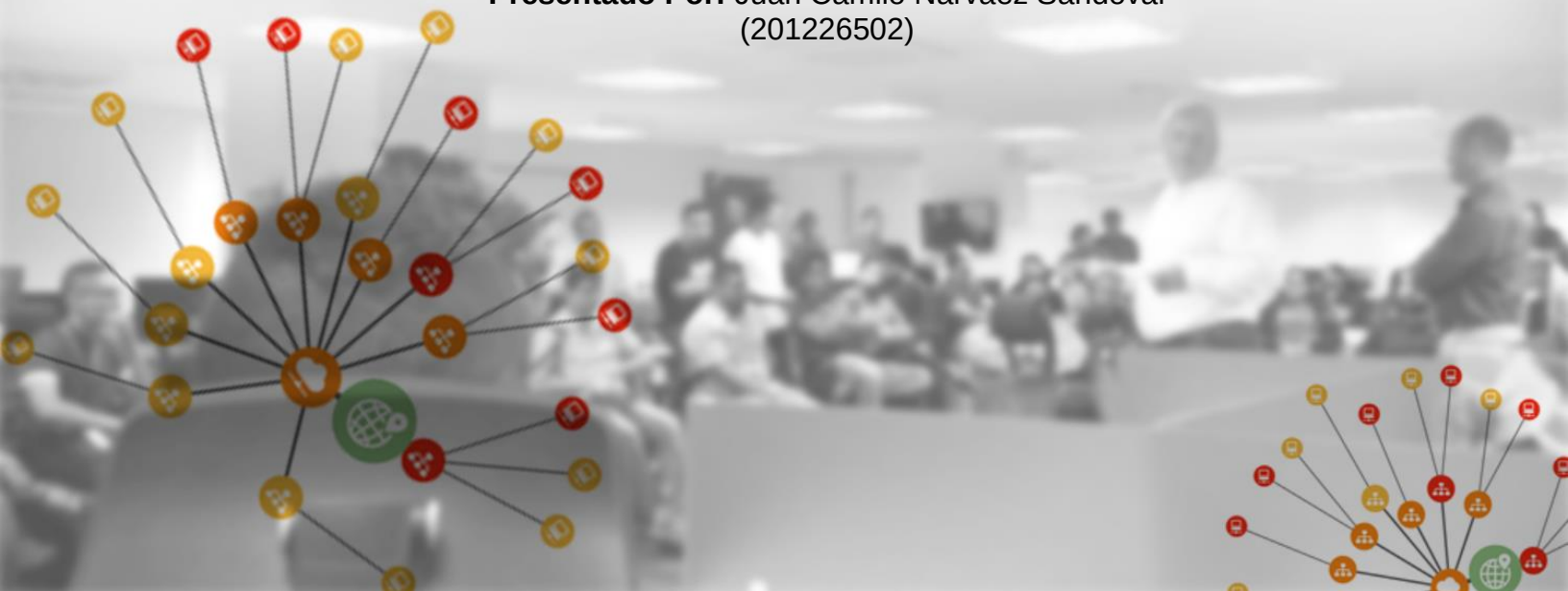
Universidad del Valle

ALERT LOGIC®
Security. Compliance. Cloud.

INTERNSHIP ALERT LOGIC INC. COLOMBIA: DESARROLLO DE FUNCIONALIDADES PARA LOS PRODUCTOS CLOUD DEFENDER Y CLOUD INSIGHT

Trabajo de Grado Modalidad Pasantía
Programa Académico de Ingeniería de Sistemas

Presentado Por: Juan Camilo Narvaez Sandoval
(201226502)



INTERNSHIP ALERT LOGIC INC. COLOMBIA: DESARROLLO DE FUNCIONALIDADES PARA LOS PRODUCTOS CLOUD DEFENDER Y CLOUD INSIGHT

Modalidad Pasantía

Por:
JUAN CAMILO NARVAEZ SANDOVAL
CÓDIGO: 201226502
juan.narvaez@correounivalle.edu.co

Directores en la Empresa:
DIEGO ALEJANDRO RUIZ CAICEDO, Ing.
Ingeniero de Sistemas y Computación
Druiz@alertlogic.com

JOSE REINA, Manager
Ingeniero Electrónico y de Telecomunicaciones. M.Sc
Jmateron@alertlogic.com

Director EISC:
BEATRIZ EUGENIA FLORIAN GAVIRIA, Ph.D
Profesora Asistente EISC
beatriz.florian@correounivalle.edu.co

Facultad de Ingeniería
Escuela de Ingeniería de Sistemas y Computación
Programa Académico de Ingeniería de Sistemas
Santiago de Cali, mayo de 2018

Tabla de Contenido

Listado de Figuras

Listado de Tablas

Resumen

Este documento se condensa las habilidades aprendidas, el conocimiento adquirido y los retos enfrentados durante el periodo de 10 meses [2 del curso de la convocatoria y 8 de pasantía] en el cual se formó parte del programa de *Internship* ofrecido por Alert Logic INC Colombia a jóvenes ingenieros de sistemas que estén interesados en adquirir habilidades en tecnologías relacionadas con el desarrollo web, tales como: AngularJS, HTML5, JavaScript, Jasmine, TypeScript, GitHub entre otras más.

Durante el periodo de pasantía se trabajó principalmente en la generación de 435 pruebas unitarias, la corrección de 5 defectos y construcción de 4 componentes en AngularJS.

Los productos derivados del proceso de construcción de las funcionalidades, pruebas y corrección de defectos podrán ser accedidos exclusivamente por el director y jurados de este trabajo de grado en modalidad pasantía, dadas las cláusulas de confidencialidad de la empresa Alert Logic INC Colombia. Las respectivas capturas de pantalla o videos que evidencian el funcionamiento de las funcionalidades construidas y el cumplimento de los objetivos podrá ser observado en las siguientes secciones:

Tabla 1. Objetivos y evidencias de resultados

OBJETIVO ESPECIFICO	RESULTADO O PRODUCTO ESPERADO	UBICACIÓN EN EL DOCUMENTO DE EVIDENCIA DE RESULTADOS
1	Constancia de participación en charlas o seminarios de capacitación en las metodologías y/o tecnologías a usar, emitido por la empresa.	Capítulo 6
2	Elaboración de un mínimo de cinco pruebas unitarias por mes	Capítulo 7
3	Implementación de soluciones a defectos del producto Cloud Defender y Cloud Insight.	Capítulo 8
4	Componente desarrollado en AngularJS.	Capítulo 9

Glosario

Aplicación de Única Pagina [*Single Page Application* • SPA]**:** Son aplicaciones web donde todas las pantallas que la componen se muestran en una misma página, sin necesidad recargar el navegador.

Balanceador de Carga: Un Balanceador de carga es un dispositivo al frente de un conjunto de servidores que atienden una aplicación el cual asigna o balancea las solicitudes que llegan de los clientes a los servidores.

Bibliotecas [Librerías]**:** Son un conjunto de implementaciones funcionales, codificadas en un lenguaje de programación, que ofrece una interfaz bien definida para la funcionalidad que se invoca.

Computación en la Nube [Cloud Computing]: Es un paradigma que permite ofrecer servicios de computación a través de una red, que usualmente es Internet.

Comunicación de Etiquetas Multiprotocolo [*Multiprotocol Label Switching* • MPLS]**:** Es un mecanismo de transporte de datos estándar creado por la IETF y definido en la RFC 3031.

DevOps: Es un acrónimo entre las palabras "Desarrollo" y "Operaciones". DevOps es un enfoque que promueve la colaboración entre líneas de negocio, desarrollo y operaciones de TI. Es una funcionalidad empresarial que habilita la entrega continua, el despliegue continuo y la supervisión continua de aplicaciones.[1]

Marco de Trabajo *[Framework]*: Entorno de trabajo o marco de trabajo, este es definido como un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver problemas de índole similar.

Infraestructura como servicio [*Infraestructure as a Service* • IaaS]**:** Es un tipo de modelo de negocio que brinda soporte en infraestructura a los clientes.

La grilla: La grilla es un conjunto máquinas encargadas de procesar miles y millones de datos pertenecientes a la aplicación como por ejemplo logs del sistema.

Modelo de interconexión de sistemas abiertos [*Open System Interconnection* • OSI**]:** El modelo de interconexión de sistemas abiertos (ISO/IEC 7498-1), más conocido como "modelo *OSI*", (en inglés, *Open System Interconnection*) es un modelo de referencia para los protocolos de la red de arquitectura en capas.

O3: Framework escrito en Angular 5 desarrollado por Kevin Nielsen Ingeniero Principal del equipo de interfaz gráfica (Platform) de Alert Logic.

Plataforma como Servicio [*Platform as a Service* • PaaS]**:** Es un tipo de modelo de negocio que brinda soporte en infraestructura y plataforma a los clientes.

Protocolo de Control de Transmisión [*Transmission Control Protocol* • TCP]**:** Protocolo de control de transmisión (en inglés), es uno de los protocolos fundamentales en Internet. Este protocolo garantiza que los datos serán entregados en su destino sin errores y en el mismo orden en que se transmitieron.

Protocolo de Datagramas de Usuario [*User Datagram Protocol* • UDP]**:** Es un protocolo del nivel de transporte basado en el intercambio de datagramas (Encapsulado de capa 4 o de Transporte

del Modelo OSI). Permite el envío de datagramas a través de la red sin que se haya establecido previamente una conexión, ya que el propio datagrama incorpora suficiente información de direccionamiento en su cabecera.

Protocolo Seguro de Internet [*Internet Protocol Security* • IPSec]**:** Es un conjunto de protocolos cuya función es asegurar las comunicaciones sobre el Protocolo de Internet (IP) autenticando y/o cifrando cada paquete IP en un flujo de datos. IPSec también incluye protocolos para el establecimiento de claves de cifrado.

Red de Área Amplia [*Wide Area Network* • WAN]**:** En inglés, es una red de computadoras que une varias redes locales, aunque sus miembros no estén todos en una misma ubicación física. Muchas WAN son construidas por organizaciones o empresas para su uso privado, otras son instaladas por los proveedores de Internet (ISP) para proveer conexión a sus clientes.

Red de Área Local [*Local Area Network* • LAN]**:** Es una red de computadoras que abarca un área reducida a una casa, un departamento o un edificio. Estas vinculan computadoras que se hallan en un espacio físico pequeño, como una oficina o un edificio. La interconexión se realiza a través de un cable o de ondas. [2]

Red Óptica Sincrónica [*Synchronous Optical Network* • SONET]**:** Es un estándar para el transporte de telecomunicaciones en redes de fibra óptica.

Scope: Es el ambiente, alcance o área donde una variable puede utilizarse.

Seguridad como Servicio [*Security as a Service* • SaaS]**:** Es un modelo de tercerización para la gestión de la seguridad. Por lo general se refiere a la gestión de seguridad provista de manera interna por una organización externa.

Software como Servicio [*Software as a Service* • SECaaS]**:** Es un tipo de modelo de negocio que brinda soporte en infraestructura plataforma y aplicación a los clientes.

SSH: Secure Shell, es el nombre de un protocolo y del programa que lo implementa, que sirve para acceder a máquinas remotas a través de una red. Esta nos permite copiar datos de forma segura.

TI: Tecnología de la información.

VPN: Es una red privada virtual (Virtual Private Network)

Mocks: También llamados “Objetos Simulados”, los mocks son objetos preprogramados con expectativas que conforman la especificación de lo que se espera que reciban las llamadas en un metodo, es decir, son objetos que se usan para probar que se realizan correctamente llamadas a otros métodos, por ejemplo, a una web API.

NPM: Es un manejador de paquetes que se ejecuta desde la línea de comandos y que maneja las dependencias para una aplicación. npm está escrito enteramente en JavaScript y fue desarrollado por Isaac Z. Schlueter a raíz de la frustración que experimentó mientras trabajaba con CommonJS

1. Introducción

Desde el surgimiento de la Ingeniería de Sistemas a partir de la fusión de la Ingeniería de Transmisión y la Ingeniería de Comunicación en 1943, y de su posterior expansión por el mundo en 1970, la Ingeniería de Sistemas se ha encargado de estudiar y comprender la realidad, con el propósito de implementar u optimizar sistemas complejos. A través del tiempo esta disciplina ha ido creciendo, obligando así, a los profesionales a fines a esta rama de la ingeniería y a muchas de las entidades que hacen uso de sus servicios, a mantenerse al tanto de los avances en cada una de sus tecnologías y así satisfacer las necesidades del mercado. Desde el uso de tubos catódicos hasta la capacidad almacenar hasta 512 GB en una memoria USB en la actualidad (2018), el software al igual que el hardware ha ido evolucionando y así mismo sus prácticas: Software como Servicio (SaaS), computación en la nube, pruebas de calidad y seguridad como servicio, son algunas de las novedades en el mundo de la informática que surgieron con el tiempo y de las que hablaremos en este documento.

La computación en la nube es un modelo de prestación de servicios de negocio y tecnología, para la gestión de los recursos de información que se almacenan de manera permanente en servidores de Internet. Este modelo permite incluso al usuario acceder a un catálogo de servicios estandarizados y responder con ellos a las necesidades de su negocio, de forma flexible y adaptativa, en caso de demandas no previsibles o de picos de trabajo, pagando únicamente por el consumo efectuado, relacionándose así fuertemente al surgimiento del concepto Software como Servicio. La adquisición de Software como Servicio en las diferentes compañías alrededor del mundo ha ido incrementando paulatinamente en los últimos años, las empresas que ofrecían sus servicios de software dando a sus clientes instaladores y complementos de hardware que pasaban a ser propiedad del cliente luego de la compra, han tenido que recurrir a migrar sus tecnologías al modelo *SaaS* para poder ser competitivos en el mercado tecnológico, aumentando así: el ahorro en costos, tiempo y mantenimiento de equipos en los que se encuentran instalados estos servicios.

La mudanza a la nube, el auge de la seguridad informática y la necesidad de calidad en los productos de software por medio del uso de pruebas, son temas que se encuentran profundamente relacionados a lo que Alert Logic INC ofrece como compañía. Alert Logic INC. Colombia, es una empresa que proporciona seguridad gestionada como una solución de servicio, integrando tecnologías innovadoras de seguridad, análisis, evaluación y procesamiento de la información que requiere participación humana y comunicación sensible [3]. Alert Logic es una empresa pionera a nivel mundial en implementar el modelo Seguridad como Servicio dentro de su portafolio, el cual está directamente relacionado con el concepto de la arquitectura Software como Servicio *(SaaS)*[4]. Para Alert Logic es fundamental el uso de tecnologías de punta que garanticen la seguridad de los datos, por tal motivo, Alert Logic se ha dado a la importante tarea de cambiar la tecnología de sus productos permitiendo que sus clientes accedan a servicios mucho más seguros y usables. Este proceso se viene realizando en diferentes etapas o fases que tienen como objetivo migrar los módulos de sus aplicativos estrella Cloud Defender y Cloud Insight a AngularJS.

Durante la pasantía desarrollada en la empresa Alert Logic INC se tuvo participación en el equipo de Front-End (UI) en tres (3) diferentes procesos que constan en: 1) El aumento de la cobertura de

las pruebas unitarias, 2) la resolución de defectos y 3) la migración de componentes a AngularJS los cuales se encuentran directamente relacionados a cada uno de los objetivos. Como resultado tangible empresarial se logró la preaprobación de una aplicación (se aprueba para integración dado una cobertura en pruebas mínima del 65%), la corrección de 5 defectos y la migración de 4 componentes a AngularJS.

El resto de este documento se organiza de la siguiente forma. Primeramente, se encuentra una descripción breve del contexto referente a este proyecto de grado. En segundo lugar, se especifican los objetivos prometidos. En tercera instancia se clarifica el marco teórico donde se definen el modelo de negocio, el concepto de nube, las herramientas de desarrollo web Front-End, Back-End y de configuración del entorno de desarrollo que fueron utilizadas en el proceso. Seguido de la sección cinco, seis y siete que contienen una ligera descripción de los aplicativos Cloud Defender y Cloud Insight, además del análisis de toma de decisiones sobre las intervenciones requeridas en estos dos aplicativos. Después, se encuentran definidos 4 capítulos que corresponden al desarrollo de cada uno de los objetivos específicos de este trabajo de grado, y para finalizar, se presentan las conclusiones e ideas de trabajo futuro.

2. Contextualización de la empresa y proyecto

Esta sección contiene la información básica de la empresa donde se realizó esta pasantía y las razones por las cuales esta misma inició el programa de prácticas y pasantías como respuesta a su crecimiento como compañía y demanda de personal capacitado en la ciudad para desempeñar cargos afines al desarrollo web y la seguridad informatica. A continuación, se presentarán los valores organizacionales, la descripción de la empresa y la justificación de este trabajo de grado.

2.1 Valores organizacionales

A continuación, algunos de los valores organizacionales más importantes que tiene Alert Logic para ofrecer a sus clientes:

- *Devoción*: Siempre es una prioridad proporcionar una gran experiencia sus clientes. Ser sensible y tener la capacidad de adaptarse a las necesidades cambiantes que puedan presentar nuestros solicitantes. Retroalimentar con soluciones y hacer seguimiento siempre.
- *Diseño Escalable*: Habilidad para reaccionar y adaptarse sin perder calidad, o bien manejar el crecimiento continuo de trabajo de manera fluida, o bien para estar preparado para hacerse más grande sin perder calidad en los servicios ofrecidos. Crear procesos que funcionen a cada uno de sus clientes.
- *Compromiso y determinación*: Establecer metas altas y con altos estándares, dando la extra-milla en cada oportunidad que se presente. Hacer lo que se dice que se va a hacer y mantener la concentración en los objetivos señalados. Superar todos los obstáculos determinación y buena actitud.
- *Innovación sin fin*: Inventar tecnologías para solucionar problemas del mundo real. Expresar todas las ideas, esforzarse por ser excelente y resolver lo irresoluble.
- *Pensando como "Nosotros"*: Comprometerse, contribuir e inspirar.
- *Progreso Constante*: Intentar siempre ser mejor cada día en todo lo que se haga. Aprovechar las oportunidades. Ser flexible, proactivo, adaptable y diligente.

2.2 Descripción de la Empresa

Alert Logic es una compañía que ofrece soluciones de Seguridad como servicio [SaaS] combinando software basado en la nube y análisis innovadores con servicios expertos para evaluar, detectar y bloquear amenazas a aplicaciones y otras cargas de trabajo. La protección se extiende a toda la pila de componentes de aplicaciones Web e infraestructura para defenderse frente a una amplia gama de amenazas del lado del servidor, incluyendo ataques de aplicaciones web difíciles de detectar como inyección de SQL, travesía y secuencias de comandos entre sitios.

Alert Logic es una empresa norteamericana especializada en seguridad informática a la cual confían sus servicios más de 4000 compañías. Esto hace que sea necesaria una planta de

profesionales especializada en el manejo de tecnologías de desarrollo web actuales que sean capaces de competir activamente en el mercado y de suplir las necesidades de sus clientes.

Esta empresa posee más de una década de experiencia, manteniendo, perfeccionando y redefiniendo soluciones seguras y flexibles diseñadas para trabajar con proveedores de servicios en modalidad de hosting o en la nube. Los productos ofrecidos por Alert Logic están en la capacidad de ofrecer soluciones de detección de intrusiones [como inyecciones SQL], análisis de vulnerabilidad y gestión de registros que se complementan con el servicio de monitoreo, vigilancia continua 24*7 y los servicios orientación brindada por expertos de su centro de gestión de seguridad. Alert Logic, siendo uno de los líderes 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, centros de datos 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 y que ofrecen una comodidad y tranquilidad mayor al cliente. Alert Logic, fundada en 2002 tiene su sede principal en Houston, Texas, con oficinas en Seattle, Cardiff, Cali y Londres.

Tabla 2.. Información Alert Logic INC

INFORMACION GENERAL ALERT LOGIC			
Razón social de la empresa	Tipo de empresa	Número de cargos	Número de empleados
Alert Logic Inc. Colombia S.A.S NIT 900.666.093-8	Desarrollo y producción de Software (según registro único tributario)	11 cargos (distribuidos en los 64 empleados directos y 5 practicantes).	64 empleados directos (contrato a término indefinido) y 5 Practicantes en las diferentes áreas

2.2.1 Naturaleza de soluciones

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. Para brindar a sus clientes las tecnologías más innovadoras de seguridad, análisis, evaluación, procesamiento de la información para sus redes, sistemas y aplicaciones Web.

2.2.2 Capacidad

Construida a escala de servicios en la nube, la plataforma patentada de Alert Logic tiene la capacidad de almacenar peta bytes de información, analizar más de 450 millones de eventos e identificar más de 60.000 incidentes de seguridad cada mes.

2.3 Justificación

Alert Logic INC, Colombia. requiere personal altamente calificado en el uso de tecnologías de desarrollo web para llevar a cabo la migración de *Cloud Defender* (aplicación desarrollada en PHP)

hacia las tecnologías usadas en Cloud Insight, más específicamente Angular JS, esta búsqueda requiere de una gran una cantidad de tiempo y conlleva altos costos de contratación de personal con experiencia. En la mayoría de los casos cuando Alert Logic contrata a un desarrollador web, éste carece de entrenamiento en AngularJS, seguridad informática, gestión de infraestructura en la nube, entre otras. Por esta razón la empresa debe dedicar tiempo y dinero en capacitar y formar a estos profesionales para que sean aptos para las labores que la empresa desempeña. Es aquí donde el programa de pasantías toma gran importancia ya que con tres generaciones de practicantes y dos de pasantes, este programa pretende capacitar y formar estudiantes afines al área de las ciencias de la computación que estén cursando los últimos semestres de su carrera para que cumplan con el perfil requerido por la empresa, logrando así una reducción de costos y de tiempo para la contratación de personal calificado.

Dada la diferencia en tecnologías entre sus dos aplicativos estrella, Alert Logic está llevando paulatinamente a Cloud Defender a que adopte la tecnología AngularJS, realizando así, una convergencia entre los dos productos hasta unificarlos bajo la misma tecnología. Hasta ahora se ha logrado migrar exitosamente algunos módulos del producto Cloud Defender, Por tal motivo, Alert Logic necesita de los recursos suficientes a nivel de personal calificado para mantener los productos Cloud Defender y Cloud Insight, y simultáneamente llevar a cabo el rediseño, reestructuración y modernización del producto Cloud Defender hacia las tecnologías utilizadas en Cloud Insight.

Alert Logic pretende que durante el periodo de pasantía cada estudiante participe en las siguientes actividades:

- Colaborar con los equipos de arquitectos, diseñadores, desarrolladores, líderes de equipo y de producto en el diseño, mantenimiento y desarrollo de las interfaces de usuario de los productos de Alert Logic para los productos Cloud Insight y Cloud Defender de acuerdo con los requerimientos solicitados, asegurando la consistencia en cuanto a usabilidad y desempeño de cada una de las aplicaciones en las que se trabaje.
- Participar activamente en la mejora continua de la experiencia de usuario y del código existente mediante la actualización constante de conocimientos en las tecnologías Web como AngularJS, HTML, CSS, PHP, TypeScript, JavaScript entre otras.
- Formar parte del proceso de codificación de pruebas unitarias para los productos.
- Participar en el proceso de desarrollo de los productos Cloud Insight y Cloud Defender, el cual se aplicará en el marco de la metodología de trabajo para desarrollo de software SCRUM *véase sección 4.3.1*
- Participar en instalaciones de los incrementos de producto.
- Ser parte del soporte a la aplicación: creación y solución de defectos.
- Proveer retroalimentación y sugerencias que ayuden a mejorar los productos de Alert Logic.
- Usar la Metodología ágil de desarrollo de software SCRUM.

Durante este proceso el pasante se ve beneficiado en la adquisición de conocimientos como: comunicación interpersonal, trabajo en equipo, conocimientos avanzados en desarrollo de software con altos estándares de calidad, seguridad informática, desarrollo web, lo que lleva a que su perfil profesional y personal mejore notablemente.

3. Objetivos

Esta sección presenta una breve recopilación de los objetivos propuestos para esta pasantía; primeramente, se da lugar a una corta descripción del objetivo general, la cual es seguida de los objetivos específicos y finalizada con los resultados obtenidos en cada uno de los objetivos.

3.1 Objetivo General

Esta pasantía pretende que el estudiante realice un entrenamiento intensivo para familiarizarse con las técnicas, tecnologías y metodologías que se usan dentro de la empresa durante el primer mes, posterior a esto, los siguientes cinco meses implementará pruebas unitarias, para luego por un mes dar solución a defectos o bugs del producto reportados por clientes de Alert Logic además de seguir trabajando en pruebas y por último, en los dos meses restantes dedicarlos a la migración de un componente de Cloud Defender a AngularJS.

Aspectos que no se incluyen dentro de la pasantía:

- No se hará diseño a nivel de arquitectura en los productos.
- No se realizarán desarrollos de módulos completos.
- No se va a liderar proyectos.
- No se diseñarán bases de datos.
- No se harán pruebas de usabilidad, ni de rendimiento del componente desarrollado al final de la pasantía.

3.2 Objetivos Específicos

1. Constancia de participación en charlas o seminarios de capacitación en las metodologías y/o tecnologías a usar, emitido por la empresa.
2. Elaboración de un mínimo de cinco pruebas unitarias por mes.
3. Identificación del requerimiento y mapeo a su solución.
4. Componente desarrollado en AngularJS.

3.2.1 Cronograma de Actividades

A continuación, se presenta el cronograma de actividades a realizar durante el periodo de pasantía, el cual se encuentra clasificado por objetivos y su respectiva duración en meses:

Tabla 3 Cronograma de actividades

ACTIVIDAD	Meses 1	2	3	4	5	6	7	8
Entrenamiento para adquirir conocimientos de metodologías y tecnologías, tales como: SCRUM, GIT, PHP, AngularJS, Manejo de Ambientes, AWS, seguridad informática, técnicas seguras de codificación.	✘							
Implementación de pruebas unitarias que soporten los incrementos de producto.		✘	✘	✘	✘	✘		
Implementación de soluciones a defectos del producto Cloud Defender y Cloud Insight.						✘		
Migración de funcionalidades específicas del producto Cloud Defender y Cloud Insight bajo la tecnología de AngularJS.							✘	✘

3.2.2 Presupuesto

Este representa el dinero invertido en los rubros referentes a recursos humanos, hardware y software necesarios para la realización de la pasantía. A continuación, se muestra la tabla donde se calcula el valor final que fue necesario para la culminación de este trabajo de grado.

Tabla 4 Presupuesto del proyecto

RUBROS	APORTE ESTUDIANTE	APORTE EMPRESA	HORAS	DINERO POR MES	MESES
Recursos Humanos					
Diego Ruiz *Manager*	$0	$ 22.222 / Hora	20	$ 444.440	5.5
José Reina *Manager*	$0	$30.000 / Hora	20	$ 600.000	1
Juan C. Narváez *Pasante*	$0	$ 13.300 / Hora	80	$ 1’600.000	8

Recursos Hardware			
Computador *MacBook Pro*	$0	$ 4’500.000	$ 4’500.000
Recursos Software			
No aplica			
Logística			
Curso Angular			$ 7’400.000

Total	Duración: 8 Meses	$ 23’944.420

3.3 Resultados Obtenidos

En la siguiente tabla se puede apreciar un breve resumen de los resultados obtenidos en cada uno de los objetivos propuestos para esta pasantía junto con la respectiva referencia a la evidencia de cada objetivo:

Tabla 5 Evidencias de productos desarrollados

OBJETIVO	¿SE LOGRO?	RESULTADOS OBTENIDOS	Evidencia en este documento
1	✔	Certificado Angular JS Desarrollo Aplicación web “The Movie Data Base”	Capítulo 8
2	✔	435 pruebas unitarias realizadas en 18 Componentes	Capítulo 9
3	✔	5 defectos solucionados, 2 en PHP y 5 en TypeScript	Capítulo 10
4	✔	4 componentes migrados a Angular JS haciendo uso del framework O3	Capítulo 11

4. Marco de Referencia

Esta sección define los conceptos referentes a las metodologías, *frameworks*, tecnologías y lineamientos que fueron ocupados durante el proceso de la pasantía realizada en Alert Logic INC Colombia en el periodo entre septiembre 2017 y mayo 2018. Este marco de referencia se encuentra dividido principalmente en tres partes que constan de: el modelo de negocio usado por la empresa, las herramientas de desarrollo web que se utilizaron para lograr el alcance de los objetivos y la metodología con la cual se llevó a cabo cada uno de los procesos que ayudaron a culminar exitosamente este trabajo de grado.

4.1 Modelo de Negocio

Un modelo de negocio se define como el conjunto de elementos característicos de una empresa, compañía o negocio que especifica cual es: el producto, el público objetivo y la manera de generar ingresos por medio del producto y/o servicio que ofrece al mercado. Alert Logic Inc. Colombia, es una empresa que proporciona seguridad informática gestionada como una solución de servicio, integrando tecnología innovadora de seguridad, análisis, evaluación, procesamiento de la información que requiere participación humana y comunicación sensible, entregando a aquellas empresas que demanden un socio líder en seguridad en la nube, unos resultados de seguridad de la más alta calidad, reduciendo por ende su complejidad. Alert Logic tiene más de una década de experiencia como pionera y refinadora de soluciones en la nube que son seguras, flexibles y están diseñadas para trabajar con proveedores de servicios de hospedaje y en la nube. Como uno de los principales proveedores de seguridad en la nube de Estados Unidos, Alert Logic cuenta con las herramientas y la experiencia que la distingue de otras compañías de seguridad en la nube.

4.1.1 Computación en la Nube • Cloud Computing

La computación en la nube es usualmente definida como un servicio web que brinda todo lo que un sistema de almacenamiento informático de oficina ofrece en la actualidad, de modo que los usuarios pueden acceder a los servicios disponibles "en la nube" sin tener conocimientos (o, al menos sin ser expertos) en la gestión de los recursos que usan. Según el *IEEE Computer Society*, La computación en la nube es un paradigma en el que la información se almacena de manera permanente en servidores de Internet y se envía a las cachés de los solicitantes del servicio[5]. La computación en la nube se puede entender entonces como servidores encargados de atender las peticiones de los clientes en cualquier momento, de los cuales se puede tener acceso a su información o servicio desde cualquier dispositivo móvil o fijo ubicado en cualquier lugar, necesitando únicamente de una buena conexión a alguno de los proveedores de servicio de Internet que se ofrezcan en la zona.

Las aplicaciones y/o servicios que son solicitados de la nube sirven a sus usuarios desde varios proveedores de alojamiento repartidos frecuentemente por todo el mundo. Esta medida reduce los costos, garantiza un mejor tiempo de actividad y reduce la posibilidad de que los sitios web sean vulnerables a los delincuentes informáticos, a los gobiernos locales y a sus redadas policiales

permanentes. A continuación, en la *Ilustración 1* se podrá observar un esquema que representa parte del flujo de datos bidireccional que ocurren en los procesos de la computación en la nube mostrando así la variedad de dispositivos desde los cuales se puede acceder a sus servicios.

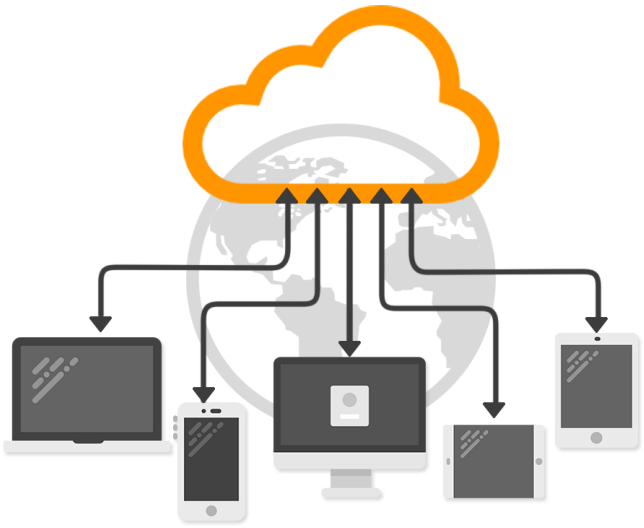


Ilustración 1 Computación en la nube

Algunos de los riesgos y oportunidades que entraña este modelo de negocio según la revista de computación PC World [6] son:

- **Privacidad de los datos** → El peligro aumenta cuando los datos se alojan en 'la nube'. En este paradigma los datos pueden residir en cualquier lugar o centro de datos, lo que supone en ciertos casos un problema legal debido a las legislaciones de muchos de los países que obligan a determinados datos a ser almacenados en territorio nacional. Sin embargo, es necesario destacar que residir información en centros de datos aumenta la seguridad de la misma si se cumple con los protocolos establecidos para el almacenamiento seguro de información.
- **Seguridad** → Es imprescindible tener la mayor seguridad ante amenazas externas y corrupción de datos, por tal motivo es importante que los proveedores de servicios garanticen transparencia, confianza y la realización de auditorías a los sistemas de información periódicamente.
- **Licencias de software** → Es preciso estudiar la compatibilidad del software bajo licencia con el software en la nube.
- **Acuerdos de nivel de servicio** → Es necesario el cumplimiento de acuerdos a nivel de servicio (SLA) antes de confiar a una empresa las aplicaciones de la misma.

Hasta hace un tiempo las aplicaciones clave de las empresas se encontraban en un servidor al final del pasillo, distribuidas a lo largo de un campus universitario o en la sede central de la misma, pero ahora con el surgimiento de la computación en la nube muchos usuarios residen sus aplicaciones de negocio, el almacenamiento, la capacidad de procesamiento y demás servicios, de manera local o en la nube. Cada día las empresas se ven más interesadas en depositar su confianza en la computación en la nube para ofrecer sus servicios y aplicaciones, pero ¿cómo se puede asegurar que dichas aplicaciones estarán seguras y tengan tan buen desempeño y disponibilidad como

cuando se encontraban en su *Red de Área Local* (LAN) o en su *Red de Área Amplia* (WAN)? Para contestar dicha pregunta, es necesario entender la nube como una manera de distribuir aplicaciones y recursos, la cual está conformada por una red que provee acceso a los servicios y da vía para que los proveedores distribuyan dichos servicios a sus usuarios.

Cuando oímos hablar acerca de la nube, en su gran mayoría se está haciendo alusión a Internet. Sin embargo, existen distintas clases de redes a las que podemos referirnos al hablar de la nube: están las nubes públicas basadas en Internet, nubes privadas que usan protocolos SONET o MPLS, e híbridas que usan por ejemplo IPSec.

A continuación, se explicará un poco cada uno de los protocolos utilizados para los distintos tipos de nubes.

Tabla 6. Protocolos de comunicación

SONET	MPLS	IPSec
Privada	Privada	Hibrida
Es un protocolo que posibilita la conexión normalizada de los sistemas de fibra óptica entre sí, aunque estos sean de distinto fabricante. Se estima que los estándares SONET/SDH podrán proporcionar las infraestructuras de transporte para la red mundial de telecomunicaciones para las siguientes dos o tres décadas (2037) SONET prácticamente solo es aplicado en Estados Unidos y Canadá	La conmutación de etiquetas multiprotocolo es un protocolo que opera entre la capa de enlace de datos y la capa de red del modelo de interconexión de sistemas abiertos OSI. Puede ser utilizado para transportar diferentes tipos de tráfico, incluyendo tráfico de voz y de paquetes IP	Es un conjunto de protocolos cuya función es asegurar las comunicaciones sobre el Protocolo de Internet (IP) autenticando y/o cifrando cada paquete IP en un flujo de datos. Los protocolos de IPSec actúan en la capa de red, la capa 3 del modelo de interconexión de sistemas abiertos (OSI). Esto hace que IPSec sea más flexible, ya que puede ser utilizado para proteger protocolos de la capa 4, incluyendo TCP y UDP.[7] IPSec también incluye protocolos para el establecimiento de claves de cifrado.

Debido a la cantidad de versiones de la nube que Internet ofrece, es evidente que en la práctica un mismo tipo de nube no satisface las necesidades y, es posible que una aplicación como el email que funciona muy bien con el desempeño variable de la nube pública no tenga problema alguno, pero para aplicativos desarrollados para los organismos de gobierno, entidades financieras y proveedores de servicios de salud, por sólo nombrar algunos, tienen reservas respecto a la privacidad y seguridad que Internet ofrece, no solo los usuarios tienen distintas necesidades, sino que las aplicaciones distribuidas a través de la nube tienen diferentes requerimientos de desempeño. Qué pasaría si es necesario el uso de la nube para la telemedicina o para el almacenamiento de datos sensibles que, su médico confiara junto con el cuidado de su salud a una red. Cuando la nube pública simplemente no es lo suficientemente segura, o cuando el desempeño es algo esencial, las instituciones financieras, los organismos de gobierno y otras empresas se

sienten más seguros con una nube privada. Cuando hablamos sobre cuestiones de privacidad y de cumplimiento, el desempeño de calidad constante y seguro es de gran importancia. A medida que más empresas migran sus funciones críticas de negocio a la nube, la elección correcta de esta se transforma en algo esencial. cuanto mejor sea la capacidad de la nube para entregar sus aplicaciones y recursos, mejor resultará la experiencia.

En conclusión, *Cloud Computing* es un modelo de prestación de servicios de negocio y tecnología, que permite al usuario acceder a un catálogo de servicios estandarizados y responder con ellos a las necesidades de su negocio, de forma flexible y adaptativa, en caso de demandas no previsibles o de picos de trabajo, pagando únicamente por el consumo efectuado, o incluso gratuitamente en caso de proveedores que se financian mediante publicidad o de organizaciones sin ánimo de lucro.

1.1.1.1 Software como servicio • Software as a Service • (SaaS)

Cuando se hace referencia al desarrollo de aplicaciones en la nube es necesario puntualizar la manera en la que se elabora, ya que dentro del concepto de nube existen distintas formas que brindan una mayor flexibilidad y sencillez a la hora de desplegar aplicaciones o mantenerlas.

Dentro de las distintas formas de servicio que puede adoptarse en la nube se encuentran: Software como servicio (SaaS), Plataforma como servicio (PaaS) e Infraestructura como servicio (IaaS), cuyas respectivas características se pueden apreciar en la *Ilustración 2* que es mostrada a continuación.

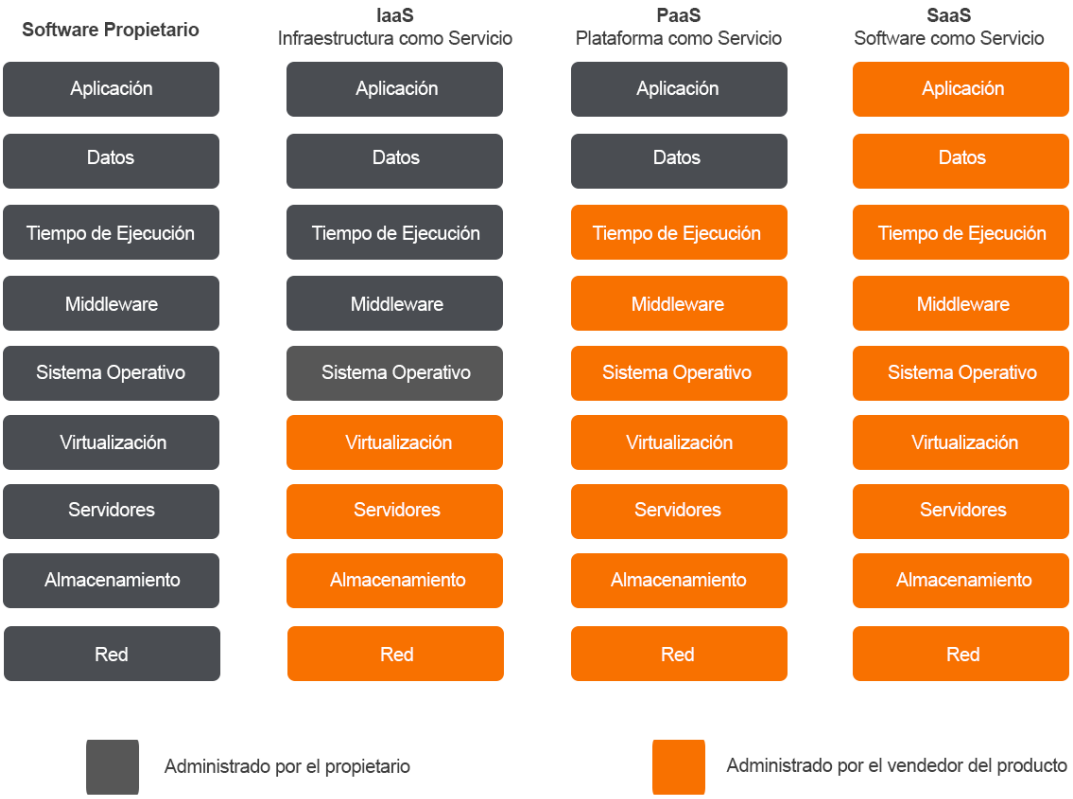


Ilustración 2. IaaS vs PaaS vs SaaS

El concepto de SaaS ha existido desde hace mucho tiempo, pero quizás en estos últimos años se ha definido claramente a qué se refiere cuando de hablar de él se trata. Básicamente SaaS es cualquier servicio basado en la web como, por ejemplo: El Web mail de Gmail, los CRM online de los bancos, las plataformas de video *streaming*, entre otros más.

En este tipo de servicios se accede a través del navegador al aplicativo sin necesidad tenerlo instalado en el hardware del propietario, siendo únicamente necesario una buena conexión a Internet para la gestión de datos.

Una definición tentativa de SaaS puede ser como un modelo de distribución de software donde el soporte lógico y los datos que manejan se alojan en servidores de una compañía de tecnologías de información y comunicación (TIC), a los que se accede vía Internet desde un cliente.

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

- **Servicios orientados al negocio:** Los cuales 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:** 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 *Ilustración 3*.

Ilustración 3. Áreas de funcionamiento del SaaS

4.1.1.1.1 Modelo de Negocio SaaS

Durante muchos años desde el inicio de la era tecnológica, muchos de los productos de software se vendieron mediante la modalidad de "software as a product", lo que significaba la compra de una licencia por parte del cliente para el uso del producto el cual es instalado en hardware propietario, convirtiéndose así el cliente en "dueño" del producto. Con el modelo SaaS en la actualidad, esta idea se torna diferente para el cliente ya que, en lugar de poseer un acceso indefinido a las funcionalidades del producto, se ofrece acceso a este mediante una suscripción que normalmente debe ser pagada mensualmente. Esto implica que, la aplicación se ejecuta en un servidor externo a la empresa reduciendo así los costos de mantenimiento, soporte y proporcionándole también una mayor estabilidad, ya que una de las características más atractivas que ofrece SaaS es que todo el desarrollo, mantenimiento, actualizaciones y copias de seguridad es responsabilidad del proveedor.

4.1.1.1.2 Arquitectura de las aplicaciones SaaS

Hay tres factores claves dentro de la arquitectura SaaS que permiten asegurar el éxito del producto: La escalabilidad, la eficiencia para atender a múltiples clientes y la configuración, de las cuales se dará a continuación una breve descripción de cada una de ellas:

- **Escalabilidad:** 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, entre otros.
- **Multi-tenant:** Los usuarios comparten en cierta manera la misma aplicación, pero esto no debe afectar el funcionamiento de sus servicios. Para hacer esto posible, se requiere de una arquitectura capaz de maximizar el intercambio de recursos a través de los usuarios, pero diferenciando los datos de cada uno de ellos. Por tanto, no se escribe 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.

Los proveedores de SaaS cada día hacen más uso de las arquitecturas multi-tenant (multi-propietario) para ofrecer sus servicios. En un entorno multi-tenant, todos los clientes y sus usuarios consumen el servicio desde la misma plataforma tecnológica.

Las arquitecturas multi-tenant se refiere a un principio en la arquitectura de software donde una única instancia del software se ejecuta en un servidor y que está al servicio de múltiples clientes.

Multi-tenant contrasta con una arquitectura multi-instancia en la cual son instancias independientes de software (o sistemas de hardware) que establecen para la organización de distintos clientes. Con una arquitectura multi-tenant, una aplicación de software está diseñada para la partición de datos y configuración de manera que cada cliente trabaja con una instancia de la aplicación virtual personalizada como se puede observar en la *Ilustración 4. Multi-tenant vs Single-tenant.* [8]

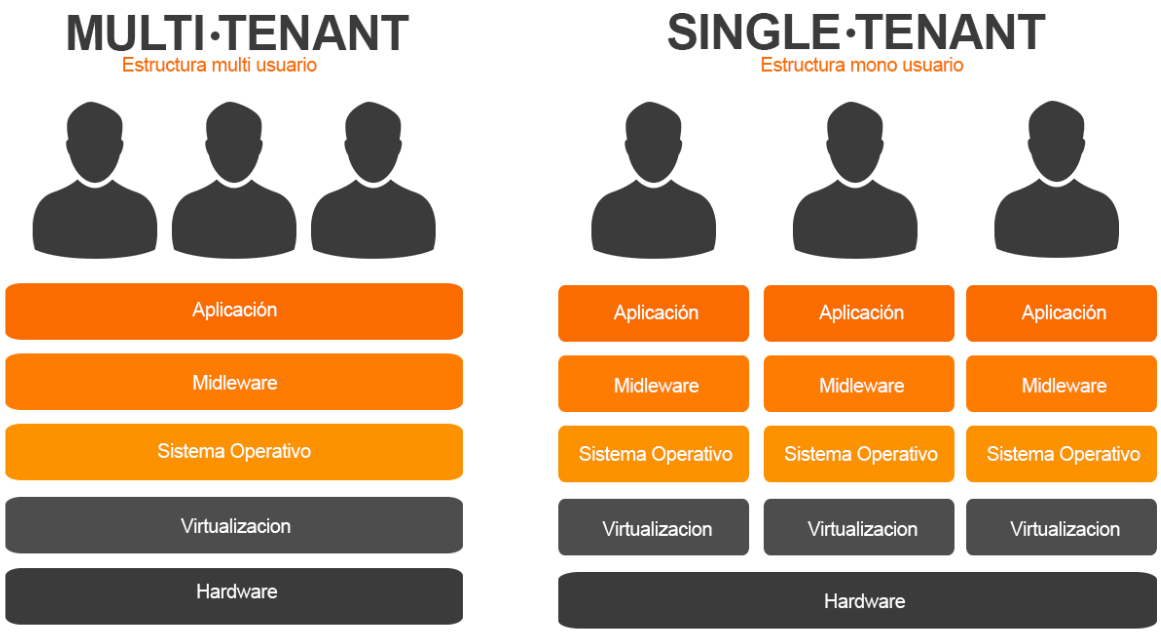


Ilustración 4. Multi-tenant vs Single-tenant

Este tipo de arquitecturas se denominan multi-tenant o multi-propietario, en ellas sobre un único recurso operan múltiples usuarios que son dueños, por así decirlo, del mismo.
Consideraciones del enfoque de esquemas compartidos:

- **Aplicación y el rendimiento de base de datos** → El rendimiento de un cliente puede verse afectada por las actividades de los clientes que comparten el servidor.
- **Seguridad** → El DBMS debe asegurar que su estructura de permisos es tal que cada cliente de datos sólo está disponible para usuarios autorizados. Además, la aplicación debe hacer uso de cualquiera de los comandos para seleccionar o restringir el esquema que se accede por un usuario.
- **Requisitos de personalización** → Cada cliente tiene su propio esquema, así que es fácil de personalizar para las diferentes necesidades que tenga cada uno de ellos.
- **El número de clientes** → Este modelo es capaz de manejar más clientes que los modelos de bases de datos separadas, pero todavía requieren una cierta cantidad de administración. La migración de los clientes a una base de datos independiente puede ser un desafío en función de las utilidades que se ofrecen por el DBMS.

Consideraciones de esquemas separados:

- **Aplicación y rendimiento de la base de datos** → El rendimiento de las consultas tendrá que ser examinados cuidadosamente para garantizar los índices adecuados existen.

- **Seguridad** → La aplicación debe utilizar código de consultas especialmente para seleccionar o restringir los datos basados en el cliente. Pruebas sólidas deben utilizarse para garantizar que un usuario no es capaz de ver los datos de los otros clientes. Encriptación de datos únicos para cada cliente no es posible.

- **Requisitos de personalización** → Todos los clientes comparten el esquema, por lo que es mucho más difícil permitir la individualización. Hay una variedad de enfoques que se pueden utilizar para proporcionar personalización. El planteamiento es permitir a un número de columnas en cada tabla genérica que se puede utilizar de diferentes maneras por cada cliente.

- **El número de Clientes** → La migración de los clientes que requieren un mejor rendimiento o la capacidad puede ser un reto, ya que los datos tendrán que ser extraídos de cada caso en operaciones separadas

4.2 Herramientas de Desarrollo Web

A continuación, se explicará un poco acerca de los distintos lenguajes, metodologías, *frameworks* y herramientas de programación de desarrollo web que fueron utilizadas para lograr los objetivos definidos en la tabla de resumen en la *Tabla 1. Objetivos y evidencias de resultados*. Primeramente, se hablará acerca de las herramientas básicas, para seguidamente pasar por el Back-End, Front-End y las herramientas de gestión de configuraciones que fueron seleccionadas.

4.2.1 Herramientas de Desarrollo Web Básicas

Esta sección hace énfasis especial primeramente en HTML como lenguaje básico para la descripción de contenido web y JavaScript para la gestión de contenido dinámico, posteriormente se hace hincapié en las herramientas añaden el estilo CSS al contenido y finalmente se da una descripción semántica sobre los elementos contenidos en el DOM de una página.

4.2.1.1 HTML

Lenguaje de Marcación de Hipertexto o *HyperText Markup Language*, mayormente conocido como HTML es un estándar a cargo del *World Wide Web Consortium* (W3C), el cual se dedicada a la estandarización de casi todas las tecnologías ligadas a la web, con énfasis especial en la escritura e interpretación del lenguaje de marcado para la elaboración de páginas web.

Desde el comienzo del uso de Internet y junto con él, el uso de HTML a través del tiempo se ha podido evidenciar un crecimiento exponencial [como se puede ver en la *Ilustración 5. Internet a través del tiempo*] y junto con esto el surgimiento de nuevas tecnologías informáticas que llevaron a HTML a realizar su primera actualización grande para HTML 4.0 en 1999 y posteriormente a su versión más actual HTML5 en 2014.

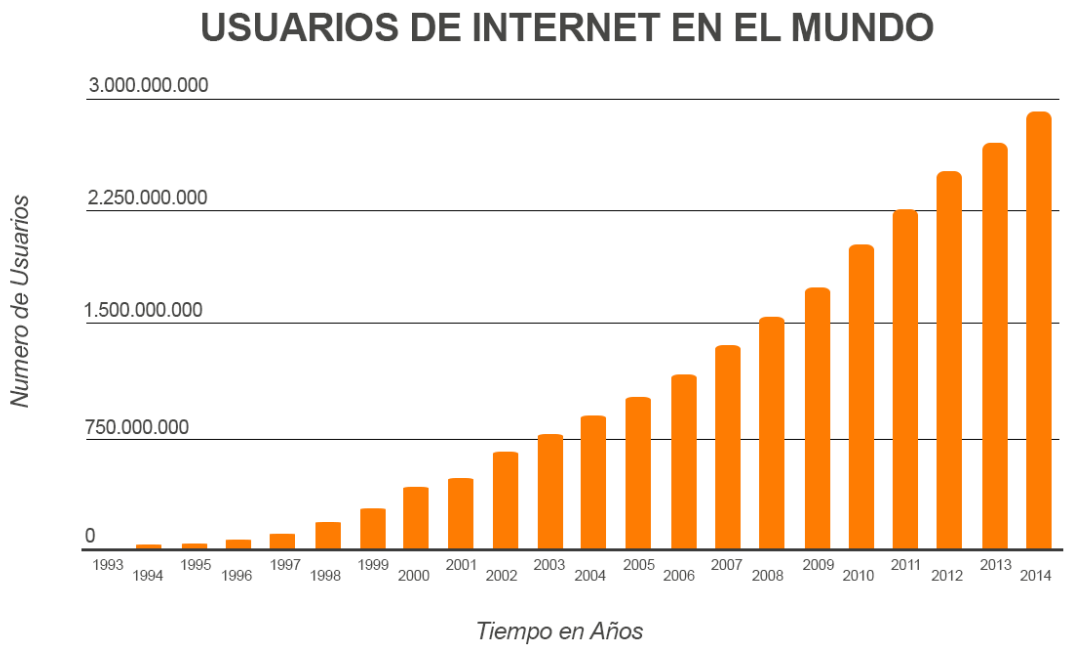


Ilustración 5. Internet a través del tiempo

Si bien HTML estuvo bien por más de una década, para el año 2014 fue necesaria una actualización. La diferencia más significativa entre las versiones más viejas de HTML y la versión 5 fueron la integración de audio y video dentro de las especificaciones del lenguaje, Adicionalmente a esto, HTML5 incluye las siguientes actualizaciones [9]:

- Los elementos obsoletos como centro, fuente y golpe fueron eliminados.
- Las reglas de análisis fueron mejoradas para permitir un análisis y compatibilidad más flexible
- Nuevos elementos que incluyen video, tiempo, navegación, sección, progreso, metro, lado y lienzo
- Nuevos atributos de entrada que incluyen correo electrónico, URL, fechas y horas
- Nuevos atributos, incluidos ‘charset’, ‘async’ y ‘ping’
- Nuevas API que ofrecen almacenamiento en caché sin conexión, soporte de arrastrar y soltar y más
- Soporte para gráficos vectoriales sin la ayuda de programas como Silverlight o Flash
- Soporte para MathML para permitir una mejor visualización de las notaciones matemáticas
- JavaScript ahora se puede ejecutar en segundo plano gracias a la API de trabajador web de JS
- Los atributos globales tales como ‘tabindex’, ‘repeat’ e ‘id’ pueden aplicarse a todos los elementos Convertir a infografía

La Ilustración 6 a continuación proporciona una descripción general de las principales funciones de HTML5 desglosadas por categoría.

Ilustración 6. Características Nuevas HTML5

4.2.1.2 JavaScript

JavaScript es un lenguaje de programación de código abierto que puede ser utilizado para la construcción de sitios Web interactivos. JavaScript es una especie de lenguaje de programación ligera, interpretado por la mayoría de los navegadores y que proporciona a las páginas web, efectos y funciones complementarias a las consideradas por el estándar HTML. Este tipo de lenguaje de programación, con frecuencia es empleado en los sitios web, para realizar acciones en el lado del cliente, estando centrado en el código fuente de la página web. Aunque este comparte muchas de las características y de las estructuras del lenguaje Java, JavaScript fue desarrollado de forma independiente. Al ser un lenguaje de programación tan popular ha sido elegido múltiples veces para ser la base de *frameworks* de programación como AngularJS, el cual es uno de los *frameworks* más utilizados actualmente como se puede ver en la *Ilustración 7* y del cual hablaremos más adelante en la *sección 4.2.2 AngularJS como Herramienta de Front•End*.

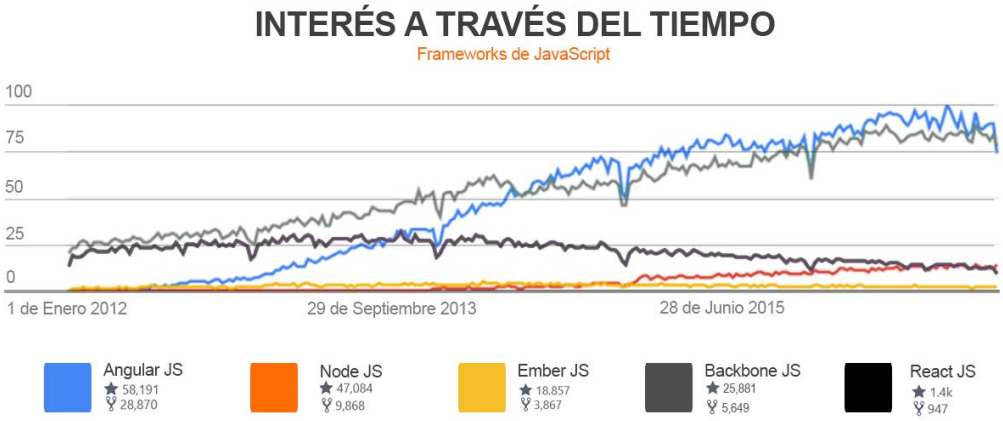


Ilustración 7. Estadísticas Angular

El lenguaje JavaScript puede interactuar con el código HTML, permitiendo a los desarrolladores web utilizar contenido dinámico. Por ejemplo, hace fácil responder a los acontecimientos iniciados por usuarios (como introducción de datos en formularios) sin tener que utilizar CGI.

4.2.1.3 CSS

CSS son las siglas de *Cascading Style Sheets*, en español Hojas de Estilo en Cascada. Esta es una tecnología desarrollada por el *World Wide Web Consortium* (W3C) que describe la presentación de los documentos estructurados en hojas de estilo para diferentes métodos de interpretación, es decir, describe la estructura de presentación de un documento en pantalla, por impresora, por voz o en dispositivos táctiles basados en Braille- Este ofrece a los desarrolladores el control total sobre estilo y formato de sus documentos. (developer.mozilla.org) [10].

En las primeras versiones del HTML, el código fuente de una página web contenía tanto la información (contenido) como la forma de representarse (diseño o formato). Hoy en día estos dos aspectos se pueden separar. La página web (El documento HTML) solo debe contener la información que se va a mostrar de cara al cliente, mientras que el formato se debe definir en las llamadas hojas de estilo (CSS) [11].

Los beneficios de usar CSS son dobles. Por un lado, se evita la construcción de archivos demasiado pesados (excluyendo el largo código requerido para las tablas anidadas y el añadido de características gráficas), y se define el "estilo visual" de un sitio entero sin necesidad de hacerlo etiqueta por etiqueta para cada una de las páginas. Esto logra que se separa hasta cierto punto la estructura (vale decir, el código) de la presentación, consiguiendo así una manera más organizada de trabajar y, lo que es más: en un sencillo documento CSS, que define lo que se podría llamar una "plantilla gráfica" para todo un sitio [12].

Actualmente las novedades que incorpora CSS3 pretenden que el diseñador tenga un mayor control sobre el diseño, dejando de recurrir a trucos que normalmente complican el código, por lo que para ello incorpora nuevas propiedades que permiten hacer algunas cosas que no se podían hacer en la versión 2, estas son algunas de ellas:

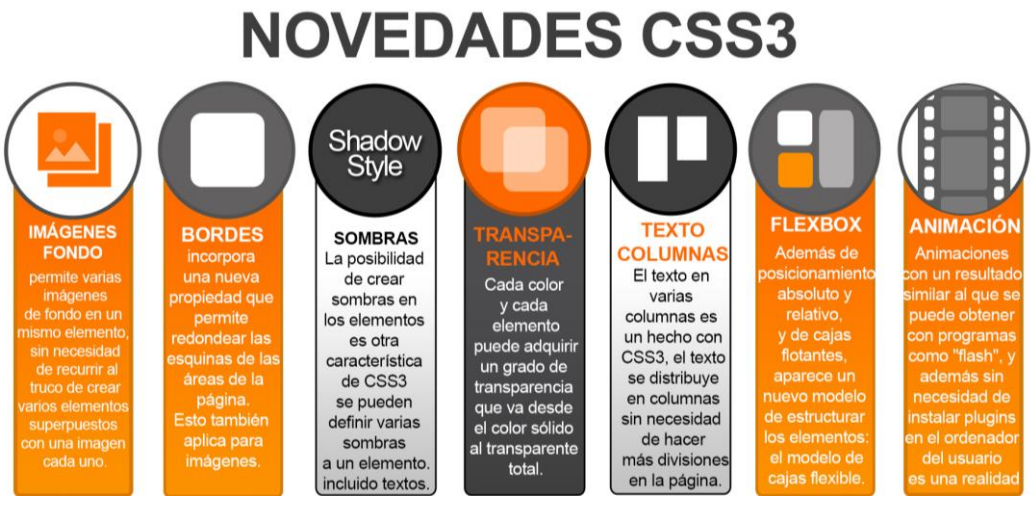


Ilustración 8. Novedades en CSS3

4.2.1.4 DOM

DOM o *Document Object Model*, es una plataforma que proporciona un conjunto estándar de objetos a través de la cual se pueden crear documentos HTML y XML, navegar por su estructura, modificar, añadir y borrar tanto elementos como contenidos. Al no apoyarse en un lenguaje de programación en particular, DOM facilita el diseño de páginas Web activas y dinámicas, proporcionando una interfaz estándar para que otro software manipule los documentos [13]. Con todo ello, cuando desarrollamos páginas web con el DOM, nos encontramos con graves dificultades para que un mismo código funcione de igual manera, por lo menos entre los navegadores más comunes [14].

Por otro lado, podemos entender también El DOM como la estructura de objetos que genera el navegador cuando se carga un documento y se puede alterar mediante JavaScript para cambiar dinámicamente los contenidos y aspecto de la página. También es posible referirse al DOM habitualmente con el nombre de jerarquía de objetos del navegador, porque realmente es una estructura jerárquica donde existen varios objetos donde unos dependen de otros. Es importante destacar ahora que como se ha mencionado anteriormente, dado que los niveles del DOM cambian de versión a versión del navegador y que las especificaciones se han entendido de manera diferente por las distintas organizaciones creadoras de los navegadores, se ha producido un marco donde trabajar con los objetos de la página difiere de un navegador a otro, lo cual puede ser un difícil de manejar cuando no se usa un *framework* adecuado que resuelva este tipo de problemas.

4.2.1.4.1 Limitaciones del Nivel 1

La especificación del Nivel 1 del DOM se limita deliberadamente a aquellos métodos necesarios para representar y manipular la estructura y el contenido de documentos. El plan para los futuros Niveles de la especificación del DOM es proporcionar:

- Un modelo de estructura para el subconjunto interno y el subconjunto externo.
- Validación contra un esquema.
- Control para representar documentos por medio de hojas de estilo.
- Control de acceso.
- Seguridad de hilos de proceso o *threads*.
- Eventos.

4.2.2 AngularJS como Herramienta de Front-End

Front-End es un término que se refiere a la separación entre la capa de presentación y la capa de acceso a datos de una aplicación, por lo que el Front-End puede entenderse como la parte del software que interactúa con el usuario.

El paso del tiempo ha transformado a la web en un medio masivo, donde millones de usuarios se congregan para consumir contenido multimedia, realizar trámites, compartir información, aprender y divertirse. Son pocos los ámbitos de la sociedad que ya no hace uso de Internet para interactuar con otras personas o incluso para trabajar. Las aplicaciones web son posiblemente una de las formas más comunes de interacción con los usuarios. Estas ya no son las páginas estáticas

de los 90’s, sino sitios que ofrecen servicios, procesan datos y realizan transacciones o cálculos complejos.

Esta sección está dedicada básicamente a AngularJS como herramienta de desarrollo Front-End. El cual es un *framework* de programación hecho en JavaScript de código abierto, soportado por Google, utilizado para crear y mantener aplicaciones web de una sola página (SPA). Su objetivo es aumentar las aplicaciones basadas en navegadores con capacidad de Modelo Vista Controlador (MVC), en un esfuerzo para hacer que el desarrollo y las pruebas sean más fáciles. Esta herramienta, contiene un conjunto de bibliotecas 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 se limita solo al envío de datos. AngularJS lee el HTML que contiene atributos de las etiquetas personalizadas adicionales, para posteriormente obedecer a las directivas de los atributos personalizados, y unir las piezas de entrada o salida de la página a un modelo representado por las variables estándar de JavaScript y TypeScript para Angular 1 y Angular 4 respectivamente. Los valores de las variables de JavaScript se pueden configurar manualmente, o recuperarse de los recursos JSON estáticos o dinámicos. A diferencia de otros *frameworks* populares, AngularJS es un *framework* estructural; no depende ni está compuesto por elementos gráficos, imágenes o CSS, solamente se enfoca en administrar la parte lógica de la aplicación.

> **Angular CLI:** Es la herramienta de línea de comandos permiten empezar a desarrollar rápidamente, añadir componentes y realizar test, así como previsualizar de forma instantánea la aplicación.

Angular es un excelente *framework* y uno de los favoritos de grandes empresas y emprendimientos precisamente por reunir las características necesarias para administrar desde un pequeño sitio web hasta aplicaciones masivas con millones de usuarios. Importantes compañías Google, Paypal, Onefootball, Microsoft Office y la IEEE con su librería digital han confiado en AngularJS como *framework* de programación web para la construcción de sus aplicaciones.

Algunas de las características que convierten Angular en una excelente opción para crear un proyecto son [15]:

- **Es extremadamente popular:** Te será muy fácil encontrar materiales, foros y contratar desarrolladores que dominen el tema.
- **No utiliza componentes gráficos:** Tienes libertad total para personalizar tu aplicación hasta el más mínimo detalle.
- **Es liviano y eficiente:** El *framework* completo pesa apenas 105kb y está optimizado para utilizar al mínimo los recursos del sistema.
- **Escribes menos código:** Todo el *framework* está diseñado para ahorrarte tiempo sin perder de vista la calidad y buenas prácticas.
- **Coexiste con otros frameworks:** Puedes utilizar AngularJS con otros *frameworks* y herramientas como jQuery, Bootstrap o PhoneGap. Sin temor a que aparezcan problemas de incompatibilidad.

En cuanto a desventajas tenemos:

- **Performance:** Angular tiene el denominado *two-way-data-binding*, que permite que las variables del modelo y la vista estén relacionadas entre sí y se actualicen en tiempo real, pero al momento de usar la aplicación hay que evaluar cada variable usada por Angular en cada *digest cycle* de la aplicación. Esto solo supone un problema potencial en la 'performance' de la aplicación si el desarrollador no lo ha tenido en cuenta y usa estructuras de datos demasiado complejas a evaluar.
- **Inyección de dependencias:** El sistema de inyección de dependencias de Angular carga todos los módulos necesarios al cargar la aplicación, esto supone que incluso si el usuario no accede a ciertas partes de esta, se cargan los módulos necesarios para que dichas partes funcionen. Esto implica un volumen de tráfico mayor del necesario, frente a otros modelos, como el implementado por RequireJS, que carga los módulos únicamente cuando estos son necesarios y se van a usar.
- **Shared state:** Angular usa los llamados *scopes* como el modelo de la aplicación. Estos se organizan en un árbol con el denominado *root scope* en la raíz que permite que por defecto cada *scope* hijo herede las propiedades de su *scope* padre y por tanto permite el acceso del hijo hacia el padre. Además, todos los *scopes* pueden acceder directamente a *root scope*, lo que implica que por defecto Angular se encuentra bajo estado compartido, eso no es bueno desde el momento en que se pretendan reusar componentes en la aplicación, pues el estado de esta podría influenciar en la creación o aparición de dichos componentes, por eso interesa que la aplicación sea sin estado. Por suerte, AngularJS permite crear lo que denomina *isolated scope* (mediante el uso de directivas), que permiten crear *scopes* propios para esas directivas, de forma que este no compartirá estado con ningún otro y de este modo lograr crear un componente que se puede reusar en toda la aplicación.

4.2.3 Herramientas Back-End

En diseño de software como se mencionó anteriormente el Front-End es la parte del software que interactúa con el usuario. El Back-End es entonces la parte que procesa la entrada desde el Front-End. La división del sistema en Front-End y Back-End es un tipo de abstracción que ayuda a mantener las diferentes partes del sistema separadas. En general la idea es que el Front-End sea el responsable de recolectar los datos de entrada del usuario, transformarlos de modo que se ajusten a las especificaciones que demanda el Back-End para poder procesarlos, devolviendo generalmente una respuesta que el Front-End recibe y expone al usuario de una forma entendible para este.

Esta sección expone primeramente el gestor de base de datos que es usado en el almacenamiento de datos, para luego pasar a los protocolos de intercambio de información estructurada y luego terminar con los formatos que son utilizados por dichos protocolos para el intercambio de información y mensajes. Las descripciones son breves y concisas ya que durante el periodo de la pasantía se trabajó mayormente en el Front-End.

4.2.3.1 Arquitectura AngularJS

Antiguamente las aplicaciones web al recibir una nueva solicitud de información al servidor respondían a dichas solicitudes con páginas web enteras, lo cual resultaba en un gasto innecesario de envío de datos y tiempo de espera a la hora de navegar.

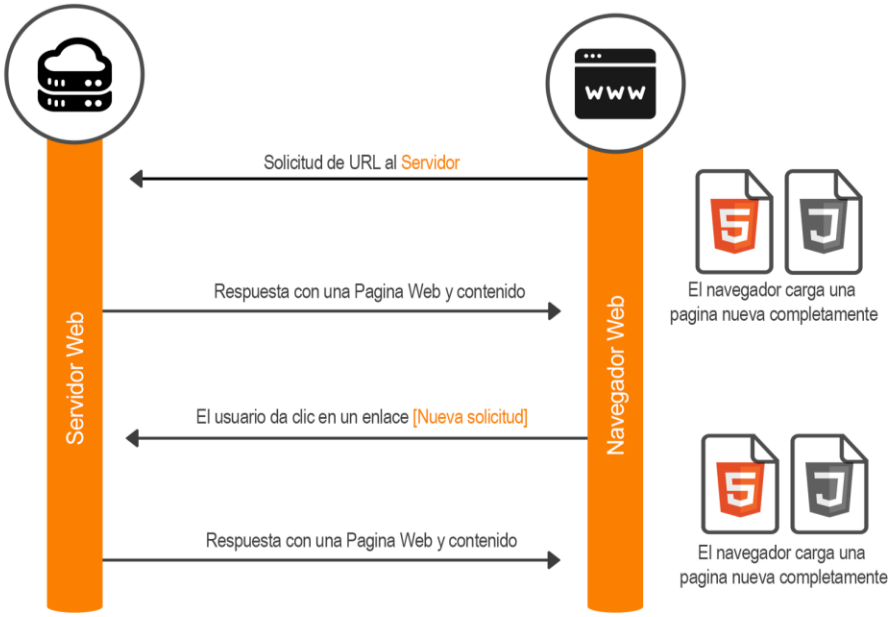


Ilustración 9. Arquitectura clásica paso de mensajes

En el caso de Angular, como se puede ver a continuación, las respuestas a las peticiones de información son atendidas únicamente con los datos que cambian y no con la construcción de la página entera, siendo así los datos cargados a los componentes JavaScript, los cuales luego son renderizados en sus respectivos HTML, mermando el tiempo de respuesta y el uso de datos por solicitud. Dichas rutas son almacenadas en un diccionario que consta de un *path* que hace referencia a la cadena de texto que es identificada en el URL y un componente en el que se asigna la vista HTML al enlace.

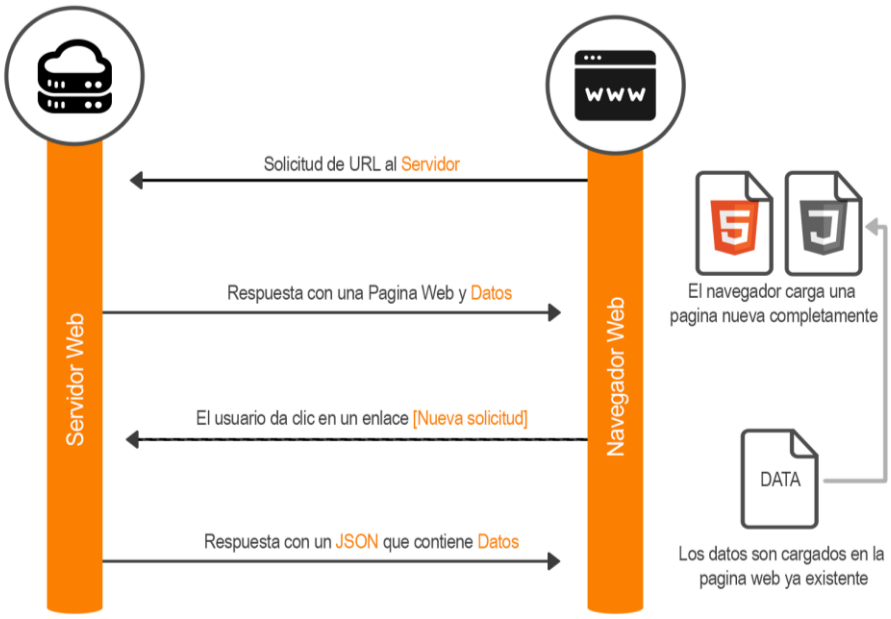


Ilustración 10. Arquitectura Angular paso de mensajes

Una de las características más apreciadas de Angular es el enlace bidireccional de datos. En el desarrollo web tradicional el browser es normalmente el encargado de los eventos a través de una propiedad llamada el *event loop*, en la que el navegador se encuentra verificando si una entrada de texto cambió, si se dio un clic, si el puntero se posicionó sobre un botón, etc. (Esto ocurre con cualquier evento que se pueda capturar tradicionalmente haciendo uso de JavaScript). Una vez que el navegador detecta cuando algo cambió, el *callback* que es registrado en el *listener* es disparado y este ejecutando asi la respectiva función. Angular extiende este *Event Loop* a través de algo que se conoce como *Angular Context*, el cual captura los eventos que permiten al *framework* actualizar la página automáticamente. En contraste con lo mencionado, mientras los eventos del *Event Loop* no actualizan la vista automáticamente, los del *Angular Context* si lo hacen, permitiendo que el DOM actualice la vista automáticamente por medio de una función llamada *Digest( )*.

La actualización automática de datos en pantalla de Angular es posible gracias a que Angular por defecto crea un observador *(watcher)* para cada atributo que pertenece a la vista, el cual es revisado constantemente para ver si algo está cambiando como ocurre en el patrón observador [16]. Cuando se hace una llamada del método *Digest( )* este itera la lista de *Watchers* para comprobar si algún valor cambio y posteriormente actualizarlo en la vista. Los eventos que *Angular Context* captura llaman a la función *Apply( ),* el cual posteriormente llama a *Digest( )* y realiza el proceso anteriormente mencionado. [17]

Haciendo una comparación con el patrón Observador, se puede apreciar que en este hay un modelo y una vista la cual depende del modelo, y que debe ser actualizada automáticamente cada vez que ocurre un cambio por medio de una lista de observadores que se encargan de actualizar sus datos como se puede ver en el esquema a continuación:

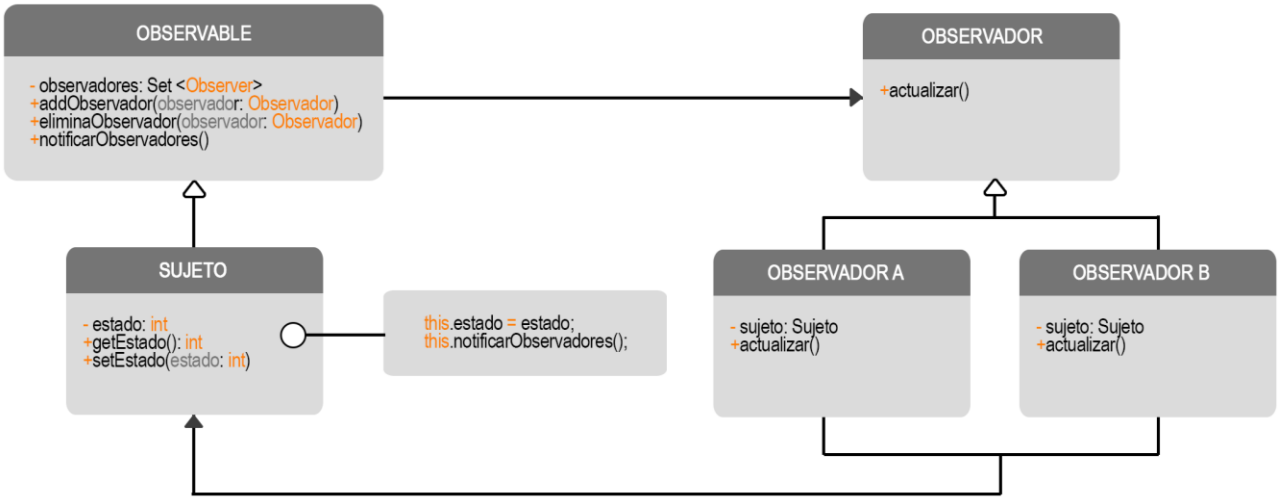


Ilustración 11. Arquitectura Observador

En el caso de Angular el modelo de datos tiene una lista de *Watchers* que son actualizados cada que un evento de *ngModel* es llamado, como el comportamiento del método *notify( )* del patrón observador. En el caso del Angular esto ocurre en el llamado de eventos como por ejemplo *ngClick*, *keyUp* en el teclado o *ngSubmit*.

4.2.3.2 MySQL

MySQL es una base de datos relacional desarrollado, distribuido y soportado por la corporación Oracle. MySQL almacena los datos en tablas separadas en lugar de poner todos los datos en un gran almacén. Esta es la base de datos de código abierto más popular del mundo; con su rendimiento, confiabilidad y facilidad de uso comprobados, MySQL se ha convertido en la principal opción de base de datos para aplicaciones basadas en la Web, utilizada por propiedades web de alto perfil como Twitter, YouTube y Facebook, entre otras más. Además, es una alternativa extremadamente popular como base de datos integrada, distribuida por miles de ISV y OEM [18].

4.2.3.3 REST

The Representational State Transfer mejor conocido como REST es una abstracción de los elementos arquitectónicos dentro de un sistema hipermedia distribuido. REST ignora los detalles de implementación del componente y sintaxis de protocolo con el fin de enfocarse en las funciones de los componentes, las restricciones a su interacción con otros componentes y su interpretación de elementos de datos significativos.

Internet es una comunicación constante entre clientes y servidores, hacer que esa comunicación sea lo más eficaz posible es el principal objetivo de muchas de las tecnologías web. REST es un estilo de arquitectura de software muy vinculado desde sus orígenes al protocolo HTTP, el cual consiste en una serie de directrices y mejoras prácticas para la creación de servicios web escalables.

Principios que definen esta arquitectura:

- Todo lo que se mueve a través de las comunicaciones de internet es un recurso, los datos se representan por el formato específico que poseen, no es un tipo de archivo y ya, JPG, HTML, XML.
- Cada recurso tiene que ser identificable por un id único, por lo cual debe haber una URI única que los diferencie de todos los demás.
- El protocolo de transmisión de datos debe usar siempre los métodos estándar que están definidos en el protocolo HTTP, donde se definen 8 acciones distintas, también conocidas como verbos: GET, POST, DELETE, HEAD, OPTIONS, TRACE, CONNECT.
- Los recursos pueden tener múltiples representaciones, de tal forma que puede ser presentado en distintos formatos, un recurso XML puede ser solicitado en formato JSON, por lo cual los recursos pueden ser presentados en un formato distinto siempre y cuando este sea soportado.
- No se pueden mantener persistencia en transacciones, el servidor trata cada petición como algo totalmente independiente, y la comunicación siempre devuelve un estado final, esto busca proteger al cliente de los cambios que ocurran en el servidor. Las API RESTFUL debe entonces ser desarrollado sin estado, ya que eso es lo que esperan los consumidores de este tipo de APIs.

RESTFUL se ha tomado fuerza últimamente tanto por la popularización de los WS y la enorme ampliación de sus usos de modo que el protocolo SOAP que funcionaba en los WS clásicos comenzara a no ser el más eficaz en muchos casos. El protocolo SOAP se basa en distintas especificaciones WS-* que aprovechan la flexibilidad de XML lo que permitió muchas especificaciones y con ello problemas posteriores.

Al utilizar REST garantizamos que los métodos HTTP son o bien seguros, lo que significa que al solicitar cualquier recurso no causa ningún tipo de cambio en su estado o bien son idempotentes como se puede observar en la *Tabla 7* , lo que significa que su respuesta es siempre la mismas sin importar las veces que sea solicitada.

Tabla 7. Métodos REST

FIABILIDAD		
MÉTODO HTTP	SEGURO	IDEMPOTENTE
GET	✔	✔
POST	✘	✘
PUT	✘	✔
DELETE	✘	✔

La idempotencia se define como la capacidad que tiene un ente (este caso el método) de realizar una misma operación varias veces, y obtener el mismo resultado que si solo se hiciese una vez.

POST no es un método idempotente ya que cada vez que es usado dicho método la respuesta del servidor varia.

4.2.3.4 SOAP

Simple Object Access Protocol o SOAP es un protocolo estándar destinado al intercambio de información estructurada en un ambiente descentralizado y distribuido. Este hace uso de la tecnología XML para definir un *framework* de mensajería expandible, proporcionando una construcción de mensaje que puede ser intercambiado a través de una variedad de protocolos subyacentes [19].

SOAP, es un lenguaje basado en XML que representa la petición y respuesta de mensajes entre servidores y clientes, fue desarrollado por Microsoft en la época de los 90 y desde el año 2000 es gestionado por la W3C. SOAP es la tecnología fundamental que subyace a todo un conjunto de especificaciones conocidas como los estándares UWS, el cual funciona como el pegamento crítico vinculante entre los juegos de herramientas de Servicios Web disponibles para muchos sistemas operativos y lenguajes de programación. SOAP no tiene como objetivo un tipo específico de proceso de negocio, a cambio es un lenguaje de programación que está diseñado para manejar los procesos y los tipos de datos que se requieran para realizar cualquier trabajo. Aunque los mensajes de SOAP se envían y reciben comúnmente a través de HTTP, también se puede usar SMTP, FTP, JMS, el servicio de mensajes de java u otros. El kit de herramientas de SOAP fue originalmente construido para lenguajes de programación apoyado por Microsoft como Visual Basic o C++ sin embargo muy pronto aparecieron kits de herramientas para otros lenguajes y plataformas.

4.2.3.5 XML • RPC

XML son las siglas en inglés de *lenguaje de marcas extensibles*, es un meta código que permite definir lenguajes de marca que se utilizan para almacenar datos en forma legible, XML proviene del lenguaje SGML y permite definir la gramática de lenguajes específicos para estructurar elementos grandes. A diferencias de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones deben comunicarse entre sí o integrar información. XML no nace únicamente para su aplicación en internet, si no que se propone como un estándar para el intercambio de información estructurada entre diferentes plataformas, se puede usar en bases de datos, editores de texto, hojas de cálculo, etc.

XML tiene un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir información de manera segura, fiable y fácil, entre sus ventajas más notables tenemos que: Es extensible, ya que después de ser diseñado un primer lenguaje de etiquetas, en post producción es posible que este sea extendido añadiendo nuevas etiquetas que sean necesarias. También su analizador es estándar, por lo cual no es necesario crear un analizador específico para cada versión del lenguaje, de esta manera se evitan bugs y se acelera el desarrollo de las aplicaciones. Mejoran la compatibilidad entre aplicaciones.

4.2.3.6 JSON

JavaScript Object Notation o Notación de Objetos de JavaScript) es un formato ligero de intercambio de datos de fácil lectura y escritura para máquinas y humanos. JSON es un formato

de texto basado en un subconjunto del Lenguaje de Programación JavaScript, Standard ECMA-262 3rd Edición - diciembre 1999 que es completamente independiente del lenguaje, pero utiliza convenciones que son ampliamente conocidos por los programadores de la familia de lenguajes C, incluyendo C, C++, C#, Java, JavaScript, Perl, Python, y muchos otros más. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos.

JSON está constituido por dos estructuras: Una colección de pares de nombre/valor. En varios lenguajes esto es conocido como un objeto, registro, estructura, diccionario, tabla hash, lista de claves o un arreglo asociativo. Y una lista ordenada de valores. En la mayoría de los lenguajes, esto se implementa como arreglos, vectores, listas o secuencias. Estas son estructuras universales; de tal forma que virtualmente todos los lenguajes de programación las soportan de una forma u otra.

4.2.4 Herramientas de Gestión de Configuración del Proyecto

Una herramienta de desarrollo de software es un programa informático que usa un programador para crear, depurar, gestionar o mantener un programa y cuya finalidad es disminuir el estrés y los tiempos de cada fase, para además mejorar los resultados obtenidos y dar mejores propuestas al cliente.

Esta sección cobija las aplicaciones, complementos y bibliotecas que ayudaron a mejorar el proceso de desarrollo del aplicativo. Primeramente, se define Git y GitHub como el controlador de versiones, para luego detallar a Jasmine como *framework* encargado de las pruebas unitarias en AngularJS y finalmente concluir hablando de Rally como gestor de tareas de SCRUM.

4.2.4.1 GIT

Git es un servicio de control de versiones diseñado para la gestión eficiente de flujos de trabajo distribuido no lineales. Git fue diseñado y desarrollado inicialmente por Linus Torvalds en 2005 a modo de herramienta de gestión de versiones 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 [20].La siguiente figura muestra los principales comandos de GIT (véase *Ilustración 12. Principales comandos de GIT*).

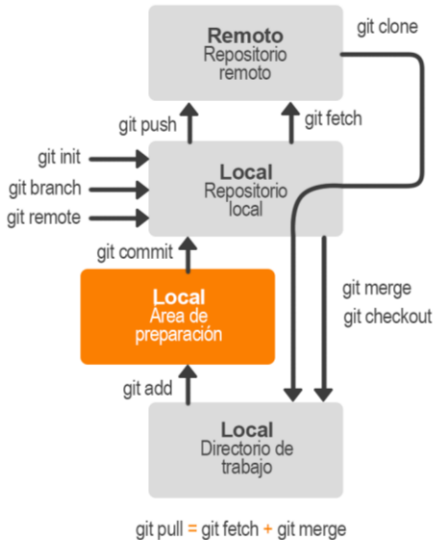


Ilustración 12. Principales comandos de GIT [20]

Características Git:

- **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, este 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.

4.2.4.2 GitHub

GitHub es una plataforma de desarrollo colaborativo de software creado en el año 2008 para alojar proyectos utilizando el sistema de control de versiones Git. El código se almacena de forma pública, aunque también se puede hacer de forma privada, creando una cuenta de pago. GitHub proporciona una interfaz Web que permite al usuario registrado crear repositorios vacíos o clonar otros repositorios hospedados 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. 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. GitHub también proporciona además del alojamiento a cada repositorio una wiki, un gestor de tareas, un completo sistema de gestión de comentarios, un cuadro de control con grafos sociales y monitoreo de trabajo[20].

4.2.4.3 Jasmine

Jasmine es un *framework* de pruebas unitarias de licencia libre para código JavaScript el cual está fuertemente influenciado por otros marcos de pruebas de unidades, como los son *Screw Unit, JSSpec, JSpec y RSpec*. [21]

Una de las características principales de las pruebas unitarias es la capacidad de ejecutarse sin necesidad de intervención manual, posibilitando así su automatización, de tal forma que las pruebas unitarias puedan repetirse tantas veces como el desarrollador quiera. Por este motivo, la velocidad con la que se ejecutan las pruebas es un factor clave a la hora de desarrollar, ya que, si dicho proceso se torna lento, este no será usado de forma habitual y se perderán los beneficios que éstas nos ofrecen.

Aunque la prueba exhaustiva del software es algo casi impracticable, las pruebas unitarias deben intentar cubrir casi la totalidad del código de nuestra aplicación, ya que el conjunto de pruebas unitarias será tan bueno como su cobertura de código.

Jasmine ofrece un amplio conjunto de combinadores en los cuales basa sus pruebas, estos poseen un valor de entrada y un valor esperado que devuelven un valor booleano el cual es verdadero si la expectativa coincide con el resultado o falso en caso de que no, logrando así una lógica muy sencilla, pero muy efectiva a la hora de probar.

4.2.4.3.1 Diseño de Pruebas con Jasmine:

Jasmine divide sus pruebas en una serie de partes denominadas *describes*, las cuales a su vez contienen sub-partes *it* que se encargan de almacenar el código pertinente a cada caso de prueba. Por lo general, cada *describe* pertenece a un método del componente a prueba, sin embargo, también en la práctica pueden resultar casos en los cuales estos se encuentran anidados debido a la complejidad de los métodos. Dentro de las características más útiles de Jasmine se encuentra que cada *describe* tienen un espacio reservado como parámetro para asignar un texto que se encarga de definir de forma general al caso de prueba en el que se está trabajando, este a su vez es complementado con los textos descriptivos más detallados que son contenidos dentro de cada uno de los *it*, los cuales se encargan de especificar un poco más detalladamente cual es la propósito de las pruebas tal como se muestra en la *Ilustración 13. Diseño de pruebas Jasmine.*

```
describe("WHEN... ",function () {
    //Define global variables

    beforeEach(function () {
        //Clear global variables
        //Set global variables
        //Set mocks
        //Set promises
        //Spies
    });

    afterEach(function () {
        //Clear variables
        //Set variables
        //Expects (toBe, ToBeNull, ToEqual, ToBeFalsy...)
        //Expects Spies (toHaveBeenCalledTimes, toHaveBeenCalledWith,
                    //.not.toHaveBeenCalled )
    });

    it("SHOULD... ", function () {
        //Clear local test variables
        //Set local test variables
        //Define promises
        //Define local Spies
        //CALLED TO FUNCTION()
        //Resolve the promise
        //Apply Scope
        //Expects (toBe, ToBeNull, ToEqual, ToBeFalsy...)
        //Expects Spies (toHaveBeenCalledTimes, toHaveBeenCalledWith,
                    //.not.toHaveBeenCalled )
    });
})
```

Ilustración 13. Diseño de pruebas Jasmine

Para finalizar esta sección es necesario nombrar a los *mocks* como figuras indispensables que permiten imitar el comportamiento o respuesta de los servicios externos a la sección de código a prueba, estos son elementos muy importantes ya que permiten incomunicar y proteger la prueba unitaria de los cambios o fallos externos ayudándonos a preservar uno de los principios fundamentales de las pruebas unitarias que es el aislamiento de código.

4.2.4.3.2 Ejecución de Pruebas en Jasmine

Luego de definir cada uno de los casos de prueba en los respectivos archivos *.spec* de cada componente, es necesario ejecutar dichas pruebas para así evaluar si el código es correcto o no. La ejecución se hace en consola por medio del comando *npm run test* el cual puede ser parametrizado en el *package.json* para que se ejecute constantemente mientras aún se programan las pruebas y así evitar tener que escribir dicho comando cada vez que se haga algún cambio, permitiendo al programador ahorrar tiempo de ejecución.

Al finalizar la ejecución de las pruebas se mostrará en consola el número de pruebas que fueron realizadas, cuáles de ellas aprobaron el test y cuáles no, ayudándose de los textos especificados en cada *describe* e *it* para asistir al programador en la tarea de ubicar el error en el caso de que al ejecutar las pruebas este ocurra.

Al final de la ejecución de las pruebas la consola debería lucir así:

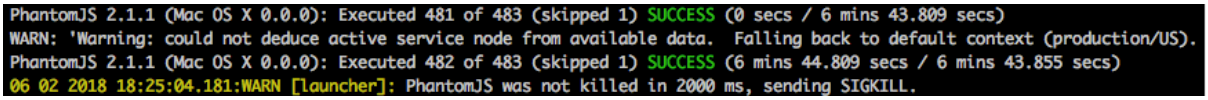


Ilustración 14. Resultado pruebas en consola

4.2.4.3.3 Reporte de Pruebas Jasmine

Para el caso del reporte, es necesario previamente haber ejecutado las pruebas para así proveer a Jasmine la información fehaciente que será posteriormente desplegada en pantalla. Dicho reporte muestra el porcentaje de cobertura con base a 4 variables: las líneas de código cubiertas, los métodos probados y los caminos que fueron testeados en cada una de las pruebas, lo cual da lugar a una buena lectura de cuál es el balance de la cobertura de los test. El uso de estas variables se debe a la complejidad que requiere el leer la cobertura de una prueba ya que por ejemplo: probar una vez todos los métodos no significa ciertamente que el código está probado al 100%, dichas funciones pueden tener múltiples caminos que esperan diferentes resultados, al igual que el tener el 100% de líneas cubiertas no significa ciertamente que todas las ramas hayan sido probadas, ya que estas se pueden enlazar entre sí y cubrir el 100% del código sin haber cubierto el 100% de los posibles casos.

Un enunciado o *statement* es una unidad sintáctica de un lenguaje de programación imperativa que expresa alguna acción que debe llevarse a cabo, un programa escrito en dicho lenguaje está formado por una secuencia de uno o más enunciados de tal forma que dicha variable nos muestra si todos los enunciados que conforman el componente han sido probados.

Tabla 8. Comparación de coberturas

Declaraciones Vs Líneas de Código	
`var x = 10; console.log(x);`	
Declaraciones = 2	Líneas de Código = 1

El comando para generar el reporte de pruebas es *npm run coverage*, al finalizar su ejecución este proporciona una URL local. La siguiente interfaz muestra una visión general del reporte de pruebas de Jasmine.

All files

77.09% Statements 2106/2732 **65.58%** Branches 663/1011 **68.4%** Functions 316/462 **76.05%** Lines 1927/2534

File	Statements		Branches		Functions		Lines	
src/incidents/evidence-list-static-entry	100%	10/10	100%	0/0	100%	2/2	100%	8/8
src	100%	54/54	100%	0/0	100%	1/1	100%	46/46
src/incidents/tutorial	100%	20/20	62.5%	5/8	100%	5/5	100%	18/18
src/environments	100%	3/3	100%	0/0	100%	0/0	100%	3/3
src/incidents	100%	41/41	50%	1/2	100%	2/2	100%	36/36
src/incidents/asset-details	100%	40/40	87.5%	7/8	100%	7/7	100%	36/36
src/incidents/audit-log	100%	26/26	100%	0/0	100%	7/7	100%	22/22
src/incidents/incidents-accordion-card	100%	28/28	100%	8/8	100%	7/7	100%	25/25
src/incidents/incident-header	100%	13/13	100%	0/0	100%	3/3	100%	11/11
src/incidents/evidence-list-aggregate-entry	100%	33/33	100%	6/6	100%	6/6	100%	29/29
src/incidents/evidence-list-expandable-entry	100%	68/68	100%	24/24	100%	12/12	100%	63/63
src/incidents/incident-topology	97.62%	41/42	100%	2/2	90%	9/10	97.44%	38/39
src/incidents/evidence-list	91.73%	122/133	55.56%	10/18	88.46%	23/26	92.13%	117/127
src/app/default-route	90.91%	20/22	66.67%	4/6	100%	4/4	89.47%	17/19
src/incidents/audit-log-entry	90%	9/10	100%	0/0	33.33%	1/3	87.5%	7/8

Ilustración 15. Reporte de pruebas en servidor local.

Aparte de esta primera pantalla del reporte, también se puede revisar cada uno de los componentes al detalle tan solo haciendo clic en su nombre. Acción nos lleva a la siguiente interfaz

All files src/incidents/summary-view

18.35% Statements 20/109 **0%** Branches 0/22 **5.26%** Functions 1/19 **16.19%** Lines 17/105

File	Statements		Branches		Functions		Lines	
summary-view.component.ts	18.35%	20/109	0%	0/22	5.26%	1/19	16.19%	17/105

Ilustración 16. Panel de prueba especifica

En esta parte de la interfaz del reporte es posible acceder a los archivos tipo TypeScript de cada componente, y observar detalladamente qué partes del código están siendo evaluadas por cada una de las pruebas y cuáles no.

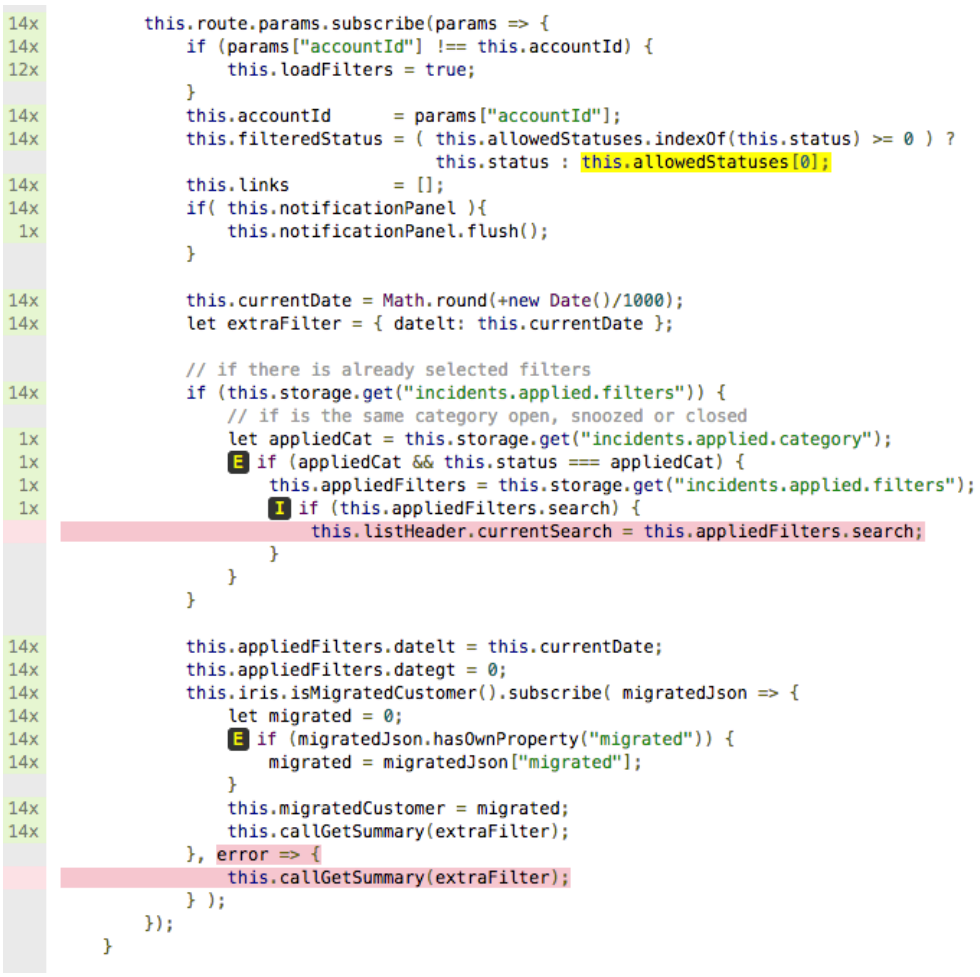


Ilustración 17. Detalle del componente probado

Como se puede observar en la figura anterior *Ilustración 17*, Jasmine también facilita la lectura de pruebas por medio de indicadores como: los números verdes de la izquierda, estos muestran cuántas veces se ha ejecutado dicha línea de código al finalizar las pruebas, los íconos negros con las letras E o I señalan las condiciones *if* o *else* que faltan por probar y, por último, resaltadas en rojo podemos observar las partes del código que no son consideradas por ninguna de las pruebas.

4.2.4.4 Rally • Herramienta de gestión del proyecto

Rally es una herramienta de software y servicios de desarrollo ágil, con ofertas que complementan y amplían las fortalezas de CA en las áreas de DevOps y *Management Cloud*. Rally ofrece características tales como personalizar tu propio tablero, priorizar la lista de características / objetivos *(Backlog),* planificar las iteraciones, añadir historias de usuario a las iteraciones y gestionar la ejecución de los ciclos, convirtiéndose así en una herramienta muy útil para gestionar proyectos usando metodologías ágiles.

Dentro de las múltiples funciones que Rally ofrece, primeramente, este permite el trabajar en múltiples proyectos los cuales pueden a su vez ser subdivididos en secciones como se muestra en

la *Ilustración 18* y así evitar tener una combinación innecesaria de tareas en un mismo *Kanban Board*.

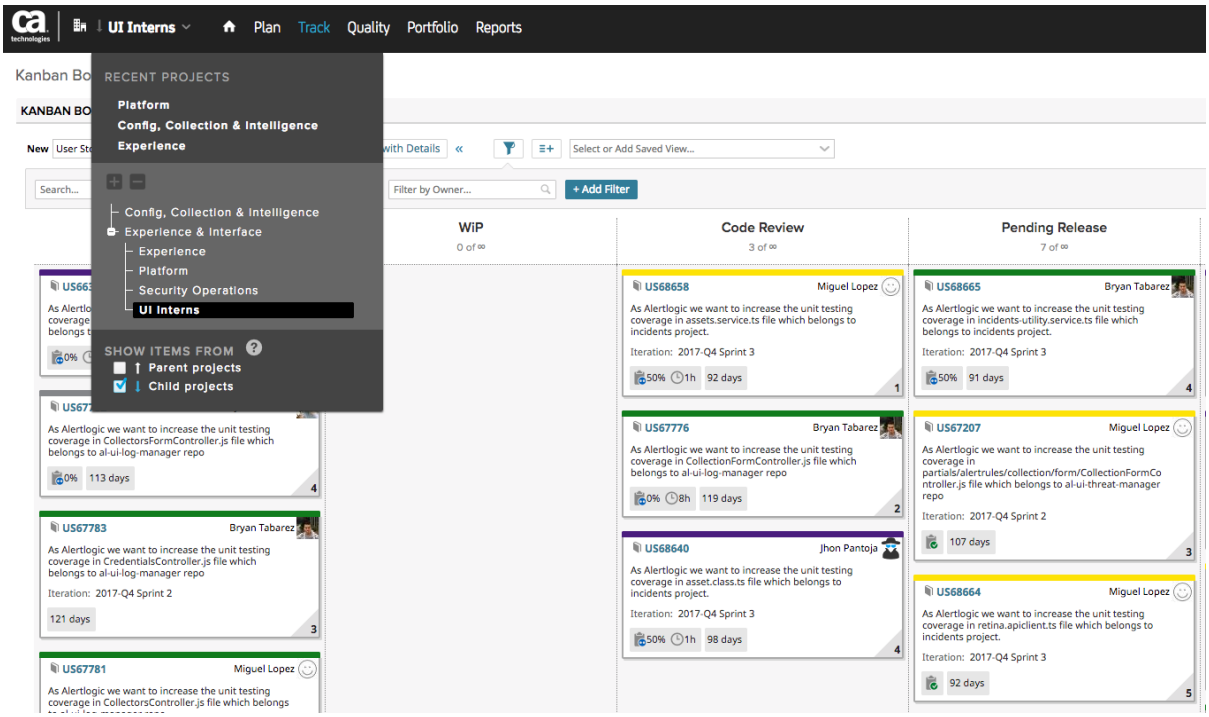


Ilustración 18. Subdivisión de proyectos y Kanban Board Rally

El *kanban board* es el tablero en el cual se organizan en forma de *postits* todas las historias de usuario estimadas a ser trabajadas durante el *sprint* actual o en futuros, dicho tablero está dividido en 5 secciones que permite monitorear en qué parte del progreso se encuentra cada una de las tareas. Dichas secciones son:

- **Sprint Backlog [**Trabajo en Reserva**]:** En esta columna se encuentran todas las tareas que han sido definidas por uno, varios o todos los miembros del equipo, las cuales cada una contiene una respectiva descripción, estimación de tiempo para desarrollo y la información adicional que pueda ser útil para la finalización exitosa de dicha tarea.
- **WiP [**Trabajo en Progreso**]:** Trabajo en progreso o *Work in Progress* (WiP) es la sección del tablero en la cual se encuentran las tareas en las que se está trabajando el equipo actualmente; al finalizar el desarrollo de la tarea, es necesario agregar los respectivos casos de prueba que ayudarán a los *code reviewers* a probar el código. Adicional a esto, también es necesario el adicionar la información que evidencie que la tarea fue finalizada exitosamente y que se cumplió con lo que se pedía en la descripción de la respectiva historia de usuario.
- **Code Review [**Código en Revisión**]:** En esta sección se encuentran las historias de usuario que se encuentran bajo la revisión de algunos de los programadores del equipo de desarrollo, estos se encargan de verificar que el código funcione y que se encuentre definido correctamente, el código es sometido tanto a correcciones de lógica como de estilo.

- **Pending to reléase [**Pendiente para Liberacion**]:** Es la sección en la que se encuentran las tareas que han pasado el filtro del *Code Review*, lo que significa que han sido aceptadas por el criterio de los otros programadores y que esperan para ser mezcladas con el código de producción.
- **Done [**Completado**]:** Es el estado en el que se encuentran las historias de usuario cuyo código ya ha sido mezclado con el código producción.

Como se pudo ver anteriormente, el *Kanban Board* está dividido en diferentes secciones que representan el estado en que se encuentra cada historia de usuario, en varios casos dicho estado no depende únicamente del programador al que se le ha asignado la historia de usuario, por lo cual cada historia de usuario tiene internamente una serie de tareas que son asignadas y que tiene 3 posibles estados que identifican el progreso de dicha tarea, por ejemplo, algunas de las tareas que puede tener usualmente una US son *Code, Code Review, Monitoring* entre otras. Esto no es un chaleco de fuerza a la hora de monitorear el proyecto, ya que dichas tareas no son definidas por Rally, sino que son dejadas al criterio de los programadores y dependen de las necesidades del equipo sobre cómo ejercer el control del progreso.

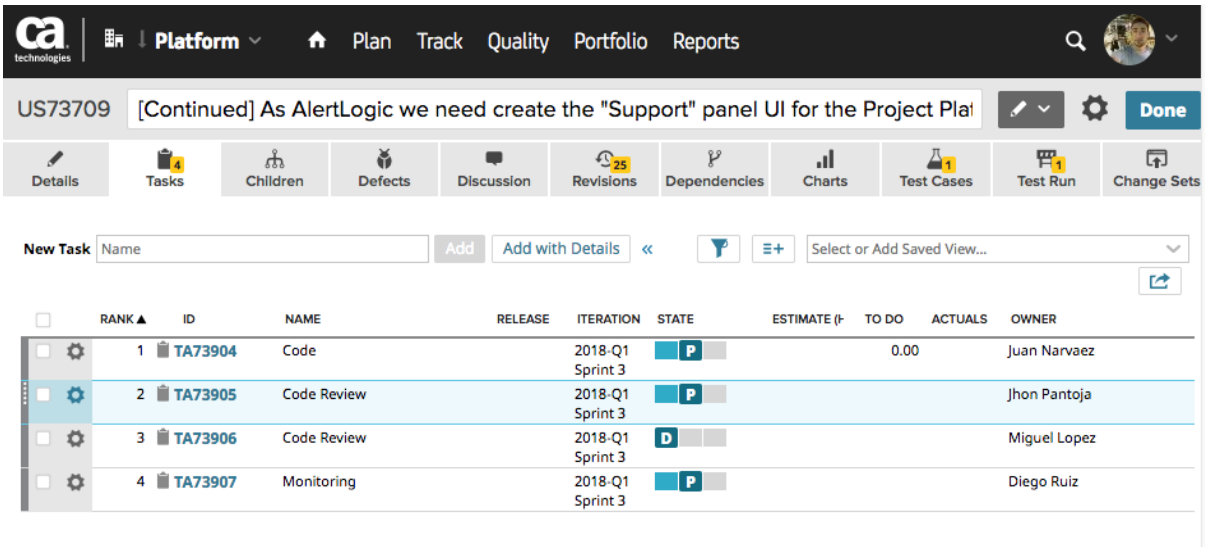


Ilustración 19. Tareas asociadas a las Historias de Usuario

Adicional a esto, las historias de usuario también cuentan con una sección de Casos de Prueba *(Test Cases)* los cuales son definidos por cada programador al terminar sus respectivas tareas, en este se deben definir el objetivo de la prueba, las precondiciones, los pasos a seguir para ejecutar el test y el resultado esperado que permita determinar si la historia de usuario fue desarrollada correctamente o si algo anda mal con respecto a lo esperado. Relacionada a esto se encuentra la sección Ejecución de Pruebas *(Test Run)* en la cual los *code reviewers* indican si la US pasó las pruebas correctamente o no como se muestra en la *Ilustración 20.*

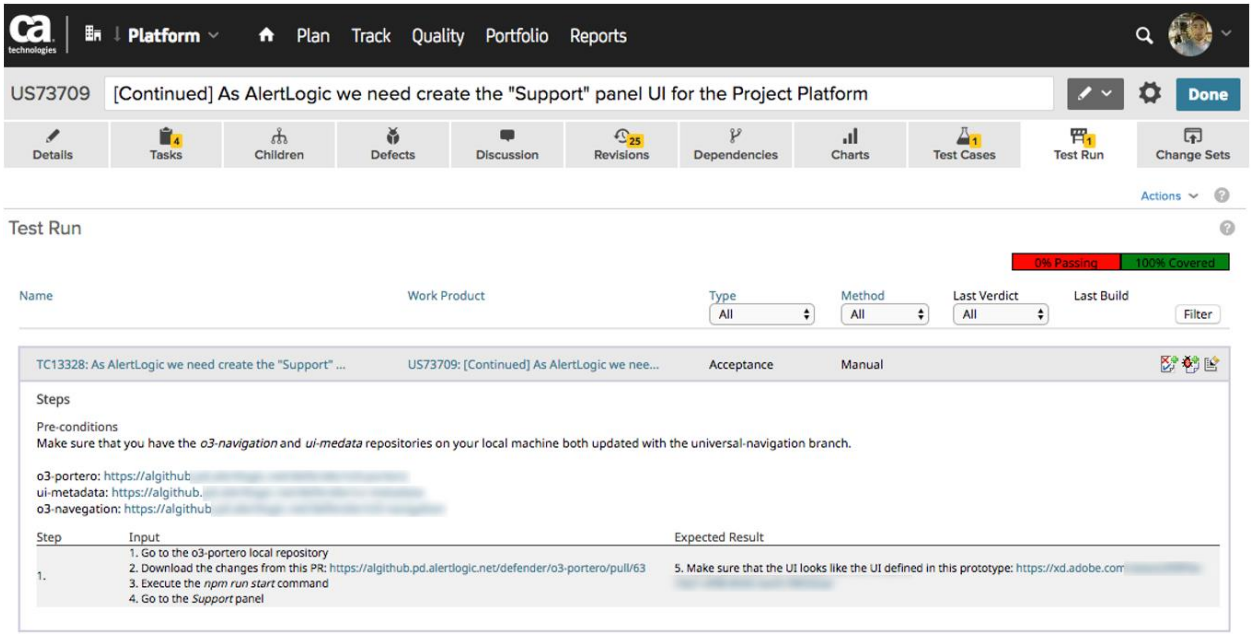


Ilustración 20. Pestaña de Test Run en Rally

Rally cuenta con una sección de discusiones *(Discussions)* en cada historia de usuario en la cual se debate y postea información importante para el desarrollo de la US, dicha sección es especialmente útil cuando de historias de usuarios relacionadas con reparación de defectos se trata, ya que aquí se define con mayor detalle el historial de pasos que debe seguirse para replicar el error.

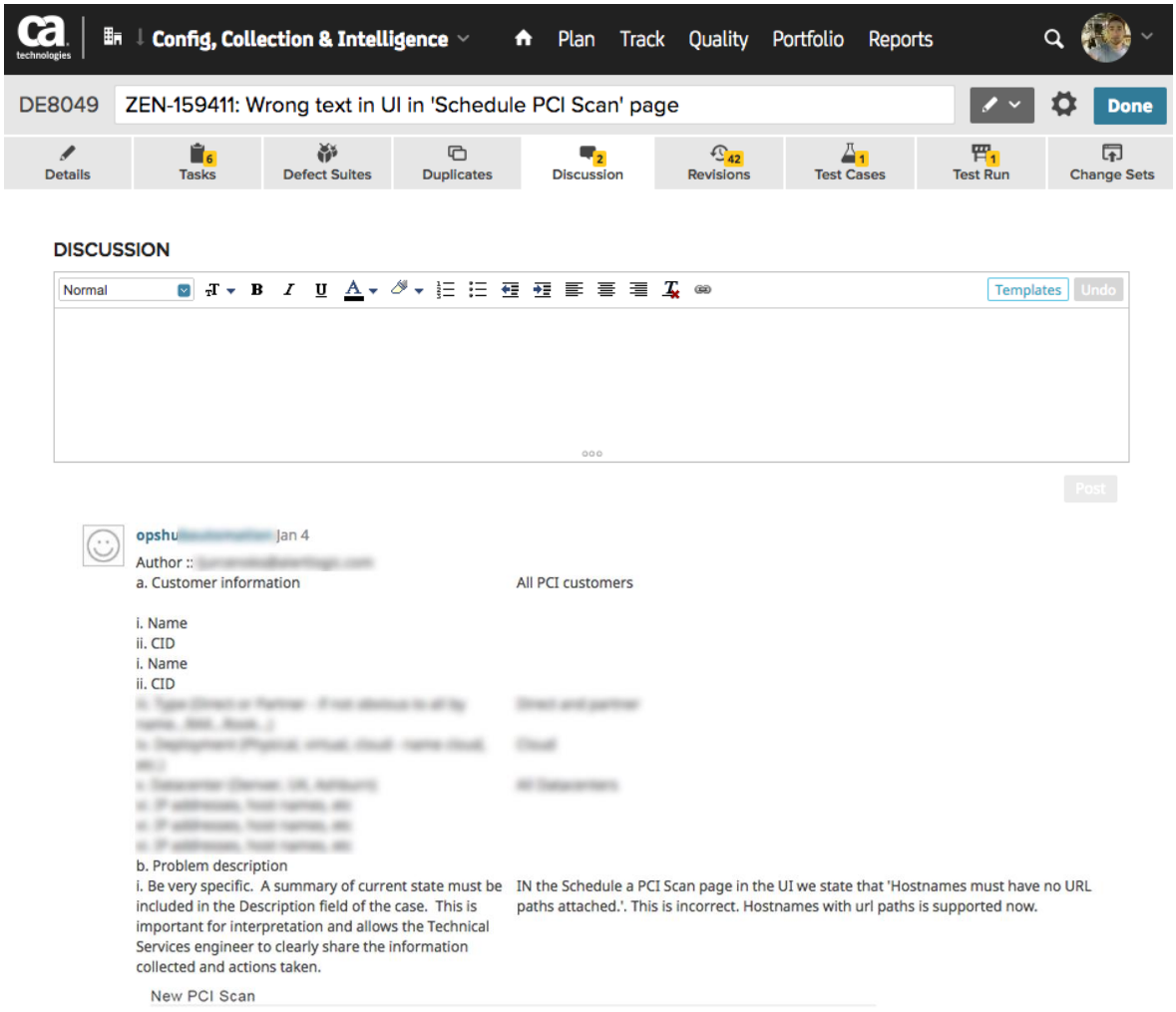


Ilustración 21. Pestaña de Discusiones en Rally

4.3 Metodología de Desarrollo Ágil

Esta sección explicará brevemente la metodología de desarrollo que fue utilizada durante el periodo en el que se realizó la pasantía, siendo SCRUM la metodología de desarrollo ágil elegida por Alert Logic para el desarrollo de sus aplicaciones, esto principalmente gracias a las entregas parciales y regulares que se deben hacer periódicamente y la flexibilidad que ofrece a la hora de lidiar con fallas o problemas inesperados.

Los procesos de desarrollo de software que buscan especificar por completo los requerimientos y, luego, diseñar, construir y probar el sistema, no están orientados al desarrollo rápido de software. A medida que los requerimientos cambian, o se descubren problemas en los requerimientos, el diseño o la implementación del sistema tienen que reelaborarse y probarse de nuevo. En consecuencia, un proceso convencional en cascada o uno basado en especificación se prolongan con frecuencia. en tanto que el software final se entrega al cliente mucho después de lo que se especificó originalmente.

En algunos tipos de software, como los sistemas de control críticos para la seguridad, donde es esencial un análisis completo del sistema, resulta oportuno un enfoque basado en un plan. Sin embargo, en un ambiente empresarial de rápido movimiento, esto llega a causar verdaderos problemas. Al momento en que el software esté disponible para su uso, la razón original para su adquisición quizás haya variado tan radicalmente que el software sería inútil a todas luces. Por lo tanto, para sistemas empresariales, son esenciales en particular los procesos de diseño que se enfocan en el desarrollo y la entrega de software rápidos.[22]

4.3.1 SCRUM

SCRUM es una metodología de desarrollo ágil creada para el desarrollo de productos de software. Esta es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto, estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos. En SCRUM se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto, por ello, SCRUM está especialmente creado para proyectos en entornos complejos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales, donde se necesita obtener resultados pronto y los requisitos son cambiantes o poco definidos [23].

SCRUM 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
- **“Sprint”.**

Para entender el ciclo de desarrollo es necesario conocer las 5 fases que definen el ciclo de desarrollo ágil [24]:

- **Concepto:** Se define de forma general las características del producto y se asigna el equipo encargado de su desarrollo.
- **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.
- **Exploración:** Se incrementa el producto en el que se añaden las funcionalidades de la fase de especulación.
- **Revisión:** El equipo revisa todo lo que se ha construido y se contrasta con el objetivo deseado.
- **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.

4.3.1.1 Artefactos de SCRUM

En el marco de trabajo SCRUM, se denomina Artefacto a aquellos elementos físicos que se producen como resultado de la aplicación de la metodología SCRUM. Los tres principales artefactos y de los cuales serán detallados en esta sección son: el Product Backlog, Sprint Backlog y el Incremento.

4.3.1.1.1 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 o en algunos casos será estimado por los miembros del equipo. 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 esta sea muy detallada pero sí lo suficiente bien definida como para que el equipo pueda trabajar. Finalmente, el Product Backlog irá evolucionando mientras el producto exista en el mercado [24]. Una herramienta muy útil para la gestión de este artefacto es Rally de la cual se habló en *la sección 4.2.4.4 Rally • Herramienta de gestión del proyecto*.

4.3.1.1.2 Sprint Backlog

Es la lista de tareas que elabora el equipo durante la planificación de un sprint a las cuales se les estipula un tiempo de finalizacion y un desarrollador a cargo de su ejecución. De esta manera, el proyecto se descompone en unidades más pequeñas donde se puede determinar o ver en qué tareas no se está avanzando. [24]

El Sprint Backlog, se muestra como una lista ordenada por prioridades para el cliente donde puede existir independencia entre una tarea u otra. Generalmente, las tareas se suelen gestionar mediante el *Scrum Taskboard*. [24]

4.3.1.1.3 Incrementos

Representa los requisitos que se han completado en una iteración y que son perfectamente operativos. Según los resultados obtenidos, el cliente puede realizar cambios cuando estos sean necesarios. [24]

Por último es necesario destacar que SCRUM también es usado para resolver situaciones en que no se está entregando al cliente lo que necesita, cuando las entregas se alargan demasiado, los costes se disparan, cuando la calidad no es aceptable, cuando se necesita capacidad de reacción ante la competencia, cuando la moral de los equipos es baja y la rotación alta, cuando es necesario identificar y solucionar ineficiencias sistemáticamente o cuando se quiere trabajar utilizando un proceso especializado en el desarrollo del producto.

5. Productos Alert Logic

5.1 Cloud Defender

Cloud Defender es un conjunto de soluciones de seguridad y cumplimiento basadas en la nube completamente administrado para la infraestructura de TI híbrida entregada como servicio. Cloud Defender agrupa y entrega los productos, servicios y características de Alert Logic en un solo paquete, es un servicio de seguridad informática que provee monitoreo constante todo el tiempo por expertos en seguridad que se encargan de realizar detección de amenazas en red, gestión de registros (logs), evaluación de vulnerabilidades y protección de aplicaciones web. Con automatización y análisis integrados se obtiene un monitoreo continuo y la inteligencia en seguridad que necesita una compañía para proteger su información donde quiera que esté. Las siguientes son características que provee este producto. [25]

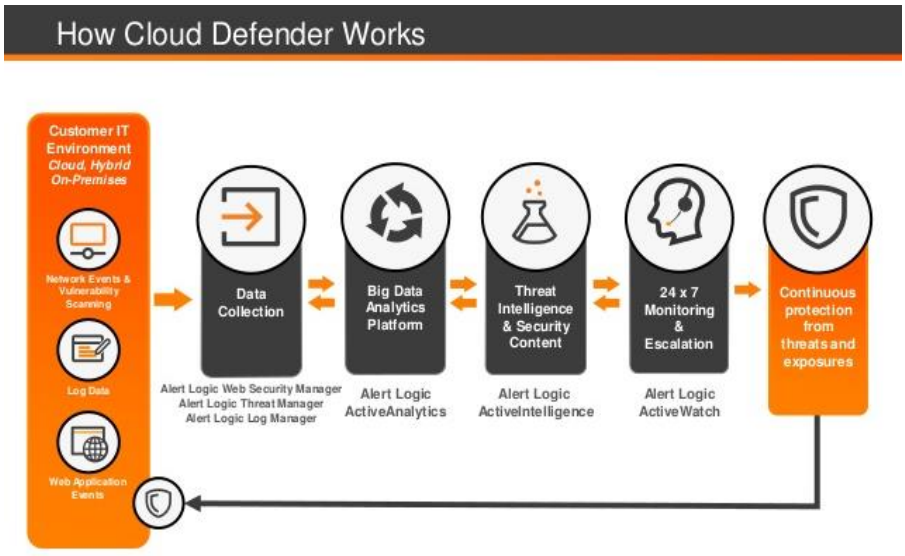


Ilustración 22. Como trabaja Cloud Defender

- **Gestión de seguridad centralizada:** Protege los datos y aplicaciones localmente, en la nube y en ambientes híbridos a través de múltiples capas, redes, sistemas y aplicaciones web.

- **Administración de detección de amenazas y respuesta:** Detección proactiva de amenazas 24x7, monitoreo en SOC (*Security Operations Center*) y rápida notificación por expertos en seguridad de GIAC (*Global Information Assurance Certification*).
- **Despliegue rápido:** Expertos se encargan del despliegue, configuración, calibración y capacitación por lo que la solución es activada rápidamente.
- **Cumple con los estándares:** Es conforme con las regulaciones de ley: PCI DSS *(Payment Card Industry Data Security Standard* • Estándar de seguridad de datos de la industria de tarjetas de pago), HIPAA (*Health Insurance Portability and Accountability Act* • *Ley de Responsabilidad y Portabilidad del Seguro de Salud*), y Sarbanes-Oxley, la cual es una Ley Norteamericana emitida en el 2002, para dar una respuesta firme a los repetidos escándalos financieros.
- **Visibilidad dentro de la infraestructura TI:** Evalúa el estado de la seguridad para entender a qué riesgos se enfrenta. 8 ley norteamericana emitida en el 2002, para dar una respuesta firme a los repetidos escándalos financieros.

5.2 Cloud Insight

Cloud Insight es una solución de administración de configuraciones y vulnerabilidades basada en la nube diseñada para ayudar a proteger las aplicaciones de negocio que se ejecutan en Servicios Web Amazon (AWS, Amazon Web Services) nativos.

Cloud Insight se integra con todas las API AWS disponibles, permitiendo la detección automática de los recursos de TI que se ejecutan en AWS, incluidos los datos de configuración de AWS y las cargas de trabajo que se ejecutan en AWS encima de ella. Cloud Insight analiza las cargas de trabajo y los datos de configuración en una biblioteca de más de 70.000 vulnerabilidades conocidas y configuraciones erróneas. Basado en este análisis, Cloud Insight presenta una lista de acciones de remediación priorizada que proporciona la información específica necesaria para reducir su riesgo de incumplimiento o compromiso. Es importante destacar que Cloud Insight proporciona una protección continua al observar constantemente su entorno y responder cuando agrega o elimina cargas de trabajo o cambia la configuración de su entorno AWS. [26]

Cloud Insight posee automatización integrada y monitoreo continuo, por tanto, provee visibilidad a través del entorno AWS identificando vulnerabilidades a medida que surgen. Entre las características de este producto se encuentran las siguientes.

- **Nativo de AWS:** Cloud Insight usa la API de AWS y datos de Cloud Trail (Servicio web ofrecido por Amazon que guarda las llamadas al API de AWS, creando archivos Log) para mantener un inventario actualizado correctamente y confiable de los activos y topologías de red. Gracias a su detección automática integrada y su capacidad de auto-despliegue, puede detectar nuevos activos que son desplegados, todo esto sin necesidad de intervención manual.
- **Protección continúa en un entorno cambiante:** Cloud Insight monitorea constantemente el entorno en búsqueda de vulnerabilidades desde la cuenta hasta el host, descubriendo automáticamente los puntos débiles en el entorno sin ningún esfuerzo por parte del cliente.

Ante los cambios en el entorno, Cloud Insight los detecta y escanea los activos nuevos, cambiados o modificados para buscar vulnerabilidades.

- **Foco en la solución y no en la vulnerabilidad:** En vez de construir una larga lista de vulnerabilidades a investigar, priorizar y determinar cómo resolver, Cloud Insight da al cliente la información que necesita para eliminarlas rápidamente y disminuir el riesgo. Esto permite que el cliente enfoque sus esfuerzos en resolver las brechas de seguridad que tienen más impacto.

5.3 Decisiones sobre los productos

Cloud Defender • Cloud Insight

A continuación, se presenta una síntesis de la entrevista realizada en mayo del 2018 a José Reina Materon, Manager del equipo de la UI, la cual tuvo como objetivo principal conocer las decisiones más relevantes que llevaron al estado actual a las aplicaciones Cloud Defender y Cloud Insight. Historia que inicia en 2015 con la liberación de Cloud Insight como aplicación y que termina con el proyecto Universal Navegación en 2018.

Finalizado 2015, Alert Logic inició el proceso de convergencia, en ese entonces, Cloud defender era una *suite* de productos como: Threat Manager, Log Manager y Scans que, inicialmente funcionaba para máquinas físicas que trabajaban sobre los data centers de los clientes. Esta arquitectura permitía ir fácilmente a la nube, la cual se convertiría posteriormente en un Frankenstein de protección física y en la nube de muchas compañías. Cloud Defender es un producto bastante antiguo y demasiado grande que contiene la suma de todos los aportes de los últimos 7 años de trabajo y el cual tuvo un papel protagónico en Alert Logic hasta la llegada de Cloud Insight en 2015. Cloud Insight como producto nativo para Amazon, fue diseñado específicamente para trabajar sobre la protección en la nube de servidores de Amazon, manejando únicamente Scans para la gestión de vulnerabilidades y remediaciones, Cloud Insight se hizo con tecnologías nuevas (más específicamente Angular 1.4 del lado de la UI, y servicios de Amazon AWS del lado del Back-End). Estas eran tecnologías totalmente nuevas para ambos frentes de desarrollo, convirtiéndose así en un producto prometedor y de un cambio visualmente radical para el cliente. La compañía tenía claro que tenía que renovar las tecnologías de Cloud Defender, por consiguiente, en un principio se pensó que era necesario reescribir todo el aplicativo, pero esto requeriría demasiado esfuerzo y un gran número de desarrolladores, por lo cual en su momento se decidió que no era un objetivo viable para la compañía en ese momento.

6 meses después de la liberación de Cloud Insight al mercado, esta no generaba los ingresos esperados, los clientes no compraban la aplicación, mientras Cloud Defender siendo un producto de interfaz y tecnologías antiguas continuaba creciendo. Este evento generó una división entre la compañía por parte de los equipos de producto y de ingeniería. Cloud Insight era un producto perfecto que no se vendía, por tal motivo, se tomó la decisión de unir estas dos aplicaciones dando inicio a la primera etapa de lo que Alert Logic llamo “Convergencia”.

El proyecto “Convergencia” consistía en empezar a migrar las tecnologías de Cloud Defender a las tecnologías de Cloud Insight, pero con la premisa de que Cloud Insight dejaría de existir como un producto individual y pasaría a ser parte de las sub-aplicaciones de Cloud Defender. En ese momento se tomó la determinación de que cada cliente que comprara Cloud Defender

automáticamente iba a adquirir Cloud Insight, lo cual era realmente útil para todos los clientes que tuvieran servicios con Amazon, además de empezar a darle visibilidad a esta aplicación.

La segunda etapa de la convergencia es el proyecto DUNCKER, este fue el primer intento de migrar las tecnologías de Cloud Defender a las de Cloud Insight, para ello se hizo un producto beta en Angular 1.4 el cual fue el primer proyecto donde trabajó la primera promoción de *interns* que resultó del programa de prácticas y pasantías. Este proyecto terminó convirtiéndose en una mezcla aun mayor de tecnologías: toda la tecnología vieja de Cloud Defender en PHP, Symfony, JavaScript y HTML, una parte de Cloud Defender migrada a Angular 1.4 conocida como Ozone y lo Nuevo de Cloud Insight quedó integrado en una sola aplicación. Ya en el 2017 empezó a hablarse acerca de un proyecto llamado 'Normandy', el cual intentó modificar toda la manera en cómo se configura Cloud Defender, objetivo que terminó realizándose por otras razones ajenas a las iniciales. Este, cambio todo el *look and feel* que era mostrado al usuario, cosa que permitió dividir la UI en diferentes subsecciones como: *overview*, *configuration*, *incidents* y otras. La organización de este producto hizo que esta fuera la nueva configuración que, no solo cambiaría a Log Manager y Threat Manager sino también Cloud Insight. Fue en ese momento cuando se unificaron totalmente, ya que antes había una separación por medio de enlaces a accesos externos, y ahora ya Cloud Insight estaba totalmente integrada dentro de la aplicación de Cloud Defender.

En la época de la primera promoción de *interns* se hablaba de convergencia y hacer que las tecnologías coexistieran juntas, hoy en día se habla de una tecnología totalmente nueva denominada O3, *framework* escrito en Angular 5 desarrollado por Kevin Nielsen Ingeniero Principal de la UI. Proyecto que tuvo el nombre de 'Universal Navigation' donde toda la aplicación se fraccionó y derivó en micro aplicaciones como: *remediation*, *overview*, *configuration*, *portero*, *search* entre otras. Logrando así una arquitectura más acorde con el modelo de MVC, dando como resultado una separación de lo visual y la lógica del negocio.

6. Apropiación de Metodologías y Tecnologías

La necesidad de siempre estar a la vanguardia en cuanto a tecnología es una de las razones más importantes por la cual Alert Logic se encuentra en un proceso de transición de hace más de 2 años en el cual apuesta por AngularJS como una tecnología ideal para manejar Front-End de sus aplicaciones.

Teniendo en cuenta lo anterior, en esta sección se hablará acerca del entrenamiento que se hizo antes y durante el periodo la pasantía. Este entrenamiento fue dividido fundamentalmente en dos grandes partes las cuales son: El curso de **96 horas** para aspirar a una vacante en el programa de *internship* Alert Logic INC Colombia, el cual comprende parte de los temas básicos con los cuales Alert Logic evalúa a sus empleados al momento de contratarlos y que fue concluido con la entrega de una aplicación web totalmente funcional desarrollada en AngularJS en su versión 2, aplicación que se usó como referente a la hora de la selección de pasantes y practicantes. Después de esto se encuentra la segunda parte, que consta de un entrenamiento de carácter laboral de **80 horas** que tuvo como principales tópicos: SCRUM como metodología de desarrollo ágil y Jasmine

como herramienta de pruebas unitarias para AngularJS, además de un breve entrenamiento en buenas prácticas en cuanto a seguridad informática laboral.

Las siguientes secciones dan detalles de cuáles fueron los temas tratados en cada uno de los cursos, las habilidades que fueron adquiridas y los entregables solicitados en el primer curso de AngularJS y en el segundo entrenamiento, finalizando con una infografía que da resumen de todo lo que trató este capítulo.

6.1 Aprendizajes en el Primer curso para Internship Alert Logic

Este curso tuvo una duración de **2 meses** con un total de **96 horas** que tuvieron lugar entre mayo 1 y junio 26 del 2017. El curso fue subdividido en 4 partes grandes partes las cuales se organizaron en el siguiente cronograma:

Tabla 9. Cronograma curso angular

Conceptos sobre RESTFUL y control de versiones			**Introducción a AngularJS**			**Enrutamiento y directivas en AngularJS**			**AngularJS mejoramiento continuo**		
16 Horas / 2 Semanas			16 Horas / 2 Semanas			16 Horas / 2 Semanas			16 Horas / 2 Semanas		
Julian Mondragón			Juan Kremer			Maryit Sanchez			David Galvis		
Mes	Dia	Clase	Mes	Dia	Clase	Mes	Dia	Clase	Mes	Dia	Clase
Mayo	Lunes 1	F	Mayo	Lunes 15	9	Mayo	Lunes 29	F	Junio	Lunes 12	25
	Martes 2	1		Martes 16	10		Martes 30	17		Martes 13	26
	Miércoles 3	2		Miérc..17	11		Miércoles 31	18		Miérc... 14	27
	Jueves 4	3		Jueves 18	12	Junio	Jueves 1	19		Jueves 15	28
	Viernes 5	4		Viernes 19			Viernes 2	20		Viernes 16	
	Sábado 6			Sábado 20			Sábado 3			Sábado 17	
	Domingo 7			Domin...21			Domingo 4			Domin...18	
	Lunes 8	5		Lunes 22	13		Lunes 5	21		Lunes 19	F
	Martes 9	6		Martes 23	14		Martes 6	22		Martes 20	29
	Miércoles 10	7		Miérc... 24	15		Miércoles 7	23		Miérc..21	30
	Jueves 11	8		Jueves 25	16		Jueves 8	24		Jueves 22	31
	Viernes 12			Viernes 26			Viernes 9			Viernes 23	32
	Sábado 13			Sábado 27			Sábado 10			Sábado 24	
	Domingo 14			Domin...28			Domingo 11			Domin...25	

En la primera sección del curso “Conceptos sobre RESTFUL y control de versiones” se hizo una breve introducción acerca de lo que es una API RESTFUL haciendo hincapié en sus principales métodos GET, POST, DELETE y UPDATE. Posterior a esto se realizó una introducción acerca de la importancia de usar alguna herramienta de control de versiones en los proyectos de desarrollo de software. Adicionalmente, se esbozó la manera en la que Alert Logic organiza sus ramas para

trabajar (este tema se detalla más adelante en esta sección en la *Ilustración 25*), después de lo cual se comenzó el entrenamiento inicial en Angular para su versión 1 denominado "Introducción a AngularJS". Este tuvo como contenido principal la configuración del entorno de desarrollo por medio de la instalación de NPM y Angular-CLI, además de la explicación, apropiación y prueba del enlace bidireccional de datos como una de las principales características de AngularJS. En el apartado denominado "Enrutamiento y directivas en AngularJS" se trató el tema de directivas como secciones de código que pueden ser reutilizadas por múltiples controladores; Las directivas son marcadores en un elemento DOM (como un atributo, nombre de elemento, comentario o clase CSS) que le dicen al compilador de HTML de AngularJS que adjunte un comportamiento específico a ese elemento del DOM. luego de esto se dio lugar al tema de enrutamiento y servicios HTTP, el cual es sumamente importante cuando de una *single page application* se trata. En la mayoría de las aplicaciones, los usuarios navegan de una vista a la siguiente a medida que realizan tareas de la aplicación, entendemos que cada vista corresponde a una URL particular, pero en una aplicación basada en Componentes como en AngularJS, cada una de estas vistas es implementada por uno o más Componentes.
Ya para finalizar el curso se escaló a la versión 2 de Angular, esto implicó la redefinición de muchos conceptos como directivas, servicios y componentes que fueron cambiados con la actualización de esta versión. Dicha transición se hizo con ayuda de la guía oficial de AngularJS *GETTING STARTED* y *TUTORIALS* disponible en la página oficial de angular: https://angular.io

El curso se concluyó con la entrega de una aplicación web totalmente funcional desarrollada en AngularJS 2 la cual tenía como objetivo principal consultar la información a mostrar de un servicio web gratuito de películas llamado *The Movie Database* [27]. Dicha información debía también ser desplegada en pantalla de una forma organizada y agradable para el usuario. Adicional a esto, se dejó a disposición de cada asistente al curso la libertad de poder adicionar funcionalidades que hicieran uso de dicha información de forma ingeniosa y atractiva. Es aquí donde la idea de hacer uso de mis conocimientos sobre minería de datos aprendidos durante el curso de KKD cobra protagonismo y se materializa en la adaptación del algoritmo de *Asociación de Datos a Priori* encargado de recomendar películas con base a una previa selección en una de las secciones del aplicativo. El resultado final de dicha entrega se puede apreciar en las capturas de pantalla:

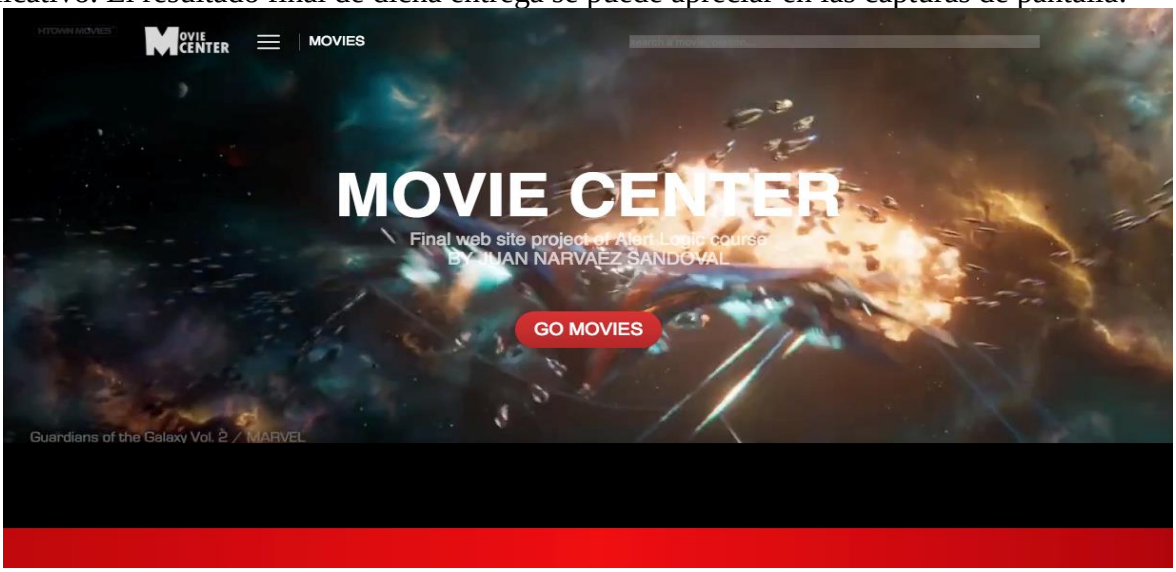


Ilustración 23. Interfaz de inicio de MOVIE CENTER

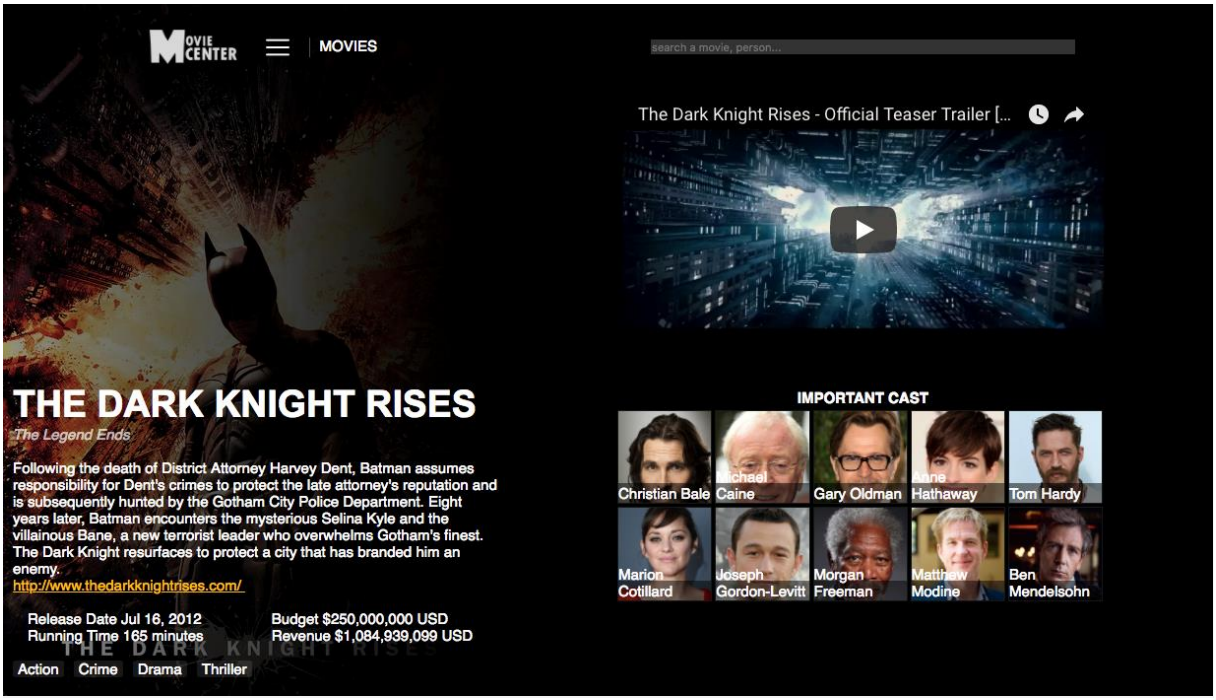


Ilustración 24. Interfaz de detalle de la película "The Dark King Rises"

Características especiales de la entrega:

- **Estilos CSS** → La maquetación de estilos se hizo 100% manual, esto posibilitó una mayor flexibilidad a la hora de plasmar la visión general del *look and feel* que se quería lograr en la página. Si bien es un proceso más tardado que el de usar ayudas como plantillas de bootstrap o material, el resultado es mucho más interesante y original.
- **Animación y transiciones** → La animación de botones y transición de imágenes usadas en la entrega se desarrollaron en su mayoría haciendo uso de una combinación de JavaScript y CSS.
- **Servicios web** → Se usó de *The Movie Database* como servicio web para la petición de información sobre las películas y sus estadísticas. Dicha información fue suministrada por el servidor en forma de archivos tipo JSON.
- **Responsivo** → Desarrollada para ser accedida desde computadores escritorio, diseño responsivo para móviles faltante.
- **Algo de minería de datos** → Para la entrega se desarrolló una variación del algoritmo de asociación de datos *A Priori*, el cual pretende a partir de la selección de un conjunto de películas, recomendar películas que sean afines a esta selección. A groso modo, el algoritmo consta de múltiples permutaciones y uniones entre los datos de películas similares proporcionados por el JSON.

Adaptación algoritmo de asociación a priori: Este algoritmo tiene como objetivo obtener conjuntos de valores de un determinado tamaño que se repiten, para luego combinarlos en reglas que cumplen un valor mínimo de soporte en el conjunto de datos seleccionado para así finalmente encontrar tendencias con base a los registros de películas similares del servicio. Para este caso en

especial se establecieron máximo 4 iteraciones ya que, si las selecciones son demasiado parecidas, el algoritmo tarda mucho en converger a causa de las uniones.

Entradas: Mínimo soporte + Conjuntos de datos seleccionados

Salida: Conjunto de datos frecuentes del tamaño más grande posible.

Desarrollo del algoritmo: Primeramente, se listan los elementos que son contenidos en cada uno de los conjuntos y se cuenta cuántas instancias hay de cada uno respectivamente, para luego comparar dicho conteo con el Mínimo soporte y descartar aquellos elementos cuyo conteo no es igual o mayores que dicho número o porcentaje. Seguidamente se pasa a hacer una unión entre cada uno de los elementos restantes para empezar una nueva iteración en la que se contarán las instancias en el conjunto de datos seleccionados de las parejas resultantes, se evalúa el conteo con el soporte mínimo, se descartan las tuplas que no cumplen dicho estamento y se pasa nuevamente a hacer la unión entre elementos, así sucesivamente hasta completar máximo 4 iteración y llegar a un conjunto de datos respuesta o retornar una lista vacía para los casos en que no se encuentren similitudes entre las películas de la selección inicial. (Asuma que los numero representan títulos de películas y que cada uno de los conjuntos de entrada son las películas relacionadas a un título en particular)

Ejemplo:
Mínimo soporte: 2, Conjuntos de datos: [1,3,4], [1,2,3,5], [2,3,5]

Iteración 1

Ocurrencias	
Set	
[1]	2
[2]	2
[3]	3
[4]	1
[5]	2

→ ✘

Ocurrencias	
Set	
[1]	2
[2]	2
[3]	3
[5]	3

→

Iteración 2

Ocurrencias		
Set		
[1,2]	1	✘
[1,3]	2	→
[1,5]	1	✘
[2,3]	2	
[2,5]	2	
[3,5]	2	

Ocurrencias	
Set	
[1,3]	2
[2,3]	2
[2,5]	2
[3,5]	2

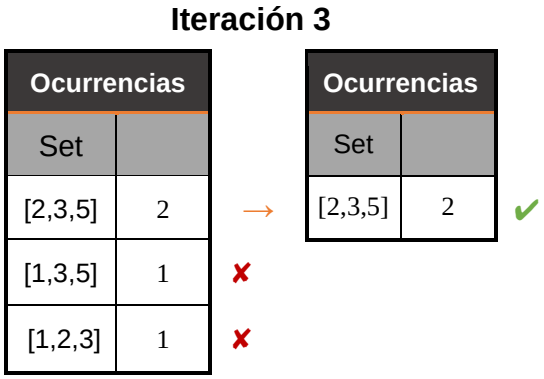


Ilustración 25. Ejemplo Algoritmo de selección a priori

6.2 Aprendizajes Durante Entrenamiento como Estudiante en Pasantía

Esta segunda parte del entrenamiento estuvo a cargo de Diego Ruiz (Manager encargado del programa de prácticas y pasantías Alert Logic) que, como SCRUM master, explicó las generalidades y particularidades del ciclo de SCRUM que se maneja en esta compañía.

A continuación, serán descritos brevemente los conceptos más relevantes y las particularidades definidas por el grupo de trabajo de Alert Logic:

- Los *Sprints* en Alert Logic tienen una duración de **2 semanas**, cerradas en las cuales se cuentan los días independientemente de si son días hábiles o no, por lo cual la aparición de festivos en el calendario no corre las fechas de entrega, pero sí afecta la carga de trabajo empleada para dicho ciclo.
- Reuniones diarias de máximo 10 minutos con el equipo comúnmente conocidas en SCRUM como *Daily scrum meetings*, en las cuales se responde las siguientes preguntas:

    - ¿Qué hice ayer?
    - ¿Qué voy a hacer hoy?
    - ¿Poseo algún bloqueo?

    Esto se hace con el fin de saber cuál es el estatus del equipo. Por lo tanto, en caso de que alguien presente algún bloqueo, este se logre identificar rápidamente y se pueda trabajar colaborativamente para resolverlo.
- Nomenclatura del sprint: Cada uno de los sprint se encuentran identificado con un código alfanumérico que consta del año, el trimestre, el número de sprint.

    Año - Trimestre - Número del Sprint

    Ejemplo de la nomenclatura:

    2018-Q2-Sprint3

- **Reunión de planeación:** En esta reunión se puntúan y asignan las historias de usuario a cada uno de los integrantes del equipo.
- **Presentación Demo:** Es aquí donde muestra el resultado de lo trabajado al final de cada sprint.
- **Reunión de retrospectiva:** Esta reunión se hace al final del sprint luego del Demo, en ella se hace la retroalimentación del trabajo realizado durante el sprint, se tratan los aspectos a mejorar para los próximos ciclos y se resaltan las buenas prácticas y logros alcanzados.

Durante esta parte del entrenamiento también tuvo lugar Rally como herramienta de gestión de Historias de usuario, la cual es esencial en la reunión de planeación de cada sprint al permitirnos visualizar cuál es el trabajo que está pendiente y en qué se está trabajando durante cada sprint. Rally asigna un código alfanumérico a cada una de sus historias de usuario dependiendo del tipo de desarrollo que su descripción contenga (Defecto o Historia de Usuario). Este código es importante ya que es usado junto con el código descriptivo del sprint para la identificación de ramas en GIT.

Los repositorios en GIT de los aplicativos de Alert Logic tiene fundamentalmente 2 ramas principales. Una de ellas la más importante llamada *producción*, en la cual se encuentra el código final que está listo para ser usado por los clientes. La segunda rama llamada *integration*, es donde es subido el código en desarrollo, el cual es probado para ver si funciona correctamente y si no tiene conflictos con el código que se encuentra actualmente corriendo. Cómo se puede ver en *Ilustración 26*, de esta rama integración se desprenden las ramas locales que usan los desarrolladores para trabajar, estas ramas normalmente tienen una nomenclatura especial (nomenclatura del sprint + el código asignado en rally para la historia de usuario) que ayuda a identificar más fácilmente cada una de las ramas.

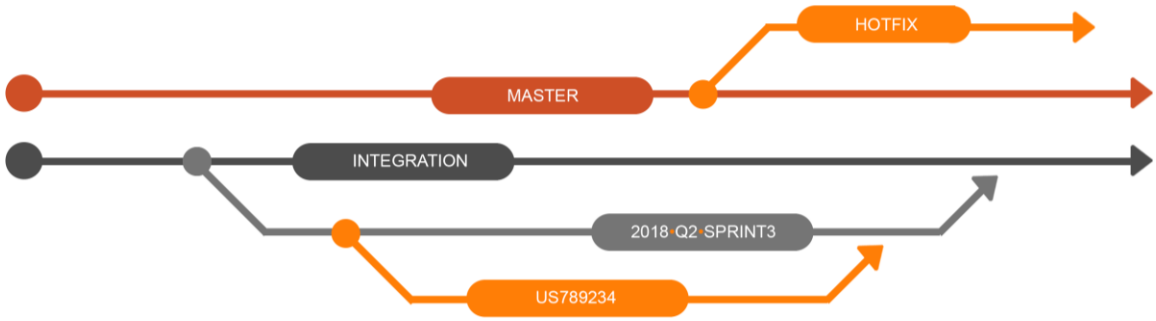


Ilustración 26. Organización de ramas GIT Alert Logic

El proceso de desarrollo ocurre de la siguiente manera: primero hay un reunión de planeación *(planning)* que se realiza con todos los integrantes del equipo, en el cual se puntúan los defectos y/o historias de usuario en las que se va a trabajar durante el sprint, por lo general cada punto representa 1 día de trabajo, por lo cual cada integrante del equipo debe ser asignado con mínimo 10 puntos por sprint (ya que cada sprint dura 2 semanas), las historias de usuario son asignadas en rally para empezar el proceso de desarrollo, a partir de aquí cada desarrollador debe crear una rama a partir de la rama integración para cada historia de usuario con el código que se le fue asignado en rally a su historia de usuario, para así empezar a desarrollar. Cuando una historia de usuario es resuelta, esta debe pasar al proceso de *pull request* y *code review* que se muestra en la *Ilustración 27*, este debe ser realizado por uno o más integrantes del equipo que estén familiarizados con el

tema, ellos deben descargar los cambios hechos a un repositorio local, y evaluar si todo funciona correctamente como está estipulado.

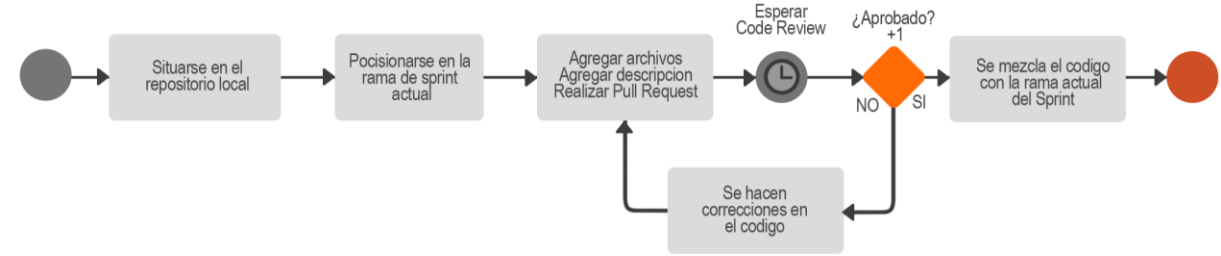


Ilustración 27. Proceso de pull request

Solo los cambios aprobados por los *code reviewers* serán mezclados en la rama de Integración, la cual posteriormente será probada en el *reléase*, después dicho código será pasado a la rama de producción.

Ilustración 28. Resumen de resultados Objetivo #1

Esta fue una etapa de superación y aprendizaje donde el mayor reto fue primeramente aplicar todos los conocimientos adquiridos durante el curso de AngularJS y en la universidad para lograr sorprender a los evaluadores del proyecto y así lograr una plaza en el programa de pasantías. La responsabilidad, perseverancia y disciplina fueron algunos de los valores que permitieron el lograr este objetivo, la presión fue alta y el resultado final excelente.

De esta etapa se aprendió la importancia del trabajo en equipo, el sentido de responsabilidad, el altruismo y lo que llama Alert Logic ‘Think we’ como valores fundamentales para la construcción de un excelente ambiente de trabajo que propicie la superación de retos en equipo.

En total fueron **96 horas** dedicadas al aprendizaje de **Angular** y **80 horas** dedicadas a **SCRUM** y **Jasmine**. Horas que concluyeron en la terminación de una aplicación web totalmente útil y en la apropiación de bases sólidas para enfrentar el siguiente objetivo que será presentado a continuación.

7. Desarrollo de Pruebas Unitarias
Cloud Defender • Cloud Insight

Las pruebas unitarias son utilizadas para comprobar el funcionamiento de una unidad de código (método, clase o servicio), esto es bastante útil a la hora de evaluar estas funcionen correcta y eficientemente por separado en un ambiente totalmente aislado en la mayoría de los casos.

Uno de los principales objetivos de las pruebas de software es el detectar defectos antes de que el producto sea instalado en el cliente o usuario final, pero es importante resaltar en este punto que, las pruebas pueden llegar a mostrar la presencia de errores, mas no la ausencia de ellos. Por lo que probar los puntos críticos del flujo de datos siempre debe ser una prioridad.

Este capítulo está asociado al segundo objetivo de este proyecto el cual define una cota inferior de 25 pruebas unitarias a ser desarrolladas en el transcurso de **5 meses**. Este mínimo fue ampliamente superado alcanzando **435 casos de prueba** como se documenta en la *Tabla 10. Resumen de pruebas unitarias a Cloud Defender y Cloud Insight.*

El criterio general para realizar las pruebas unitarias es lograr tanta cobertura como sea posible en cada componente, por lo cual no hay una prioridad latente en cuanto a que métodos se deben probar primero o cuáles deben ser ignorados. El proceso de realizar las pruebas unitarias ocurre como se ve en la *Ilustración 29.*

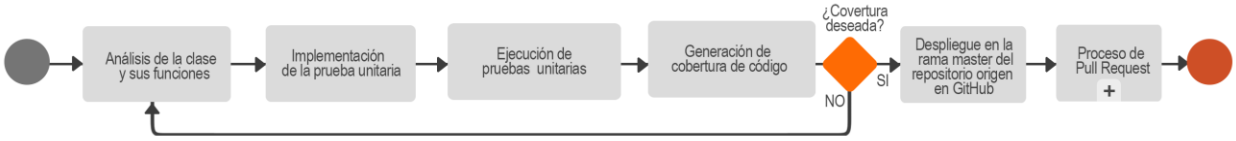


Ilustración 29. Proceso de Unit Testing

7.1 Diseño de pruebas

En el caso de Alert Logic al ser una compañía que hace uso de metodologías ágiles para llevar a cabo sus procesos de desarrollo, la etapa de diseño se ve eclipsada por el uso de principios desarrollados para la ejecución de proyectos usando metodologías ágiles como lo es F.I.R.ST el cual se encuentra definido posteriormente en esta sección. El principio fundamental de Alert Logic a la hora de ejecutar el proceso de pruebas es poder cubrir tanto código como sea posible, esto con el fin de conseguir pre aprobar cada aplicación por medio de la superación del 65% de cobertura final para la aplicación. En este orden de ideas al final de cuentas cada programador encargado de desarrollar pruebas codifica y diseña al tiempo con la idea de ahorrar horas de trabajo y lograr una

mayor cantidad de pruebas al final de cada sprint. Pruebas que apuntan al aumento en la cobertura de funciones.

FI.R.S.T: Las pruebas limpias siguen cinco reglas, cuyas iniciales forman las siglas FIRST en inglés y cuyo significado es el siguiente:

- **Rapidez:** Las reglas deben ser rápidas y ejecutarse de forma rápida. Si lo hacen lentamente, estas no se ejecutarán con frecuencia. Al no hacerlo, no se detectarán los problemas con la suficiente antelación como para solucionarlos. No se sentirá la libertad para limpiar el código, que en mayor acabará corrompiéndose.
- **Independencia:** Las pruebas no deben depender entre ellas. Una prueba no debe establecer condiciones para la siguiente. Se debe poder ejecutar cada prueba de forma independiente y en el orden que desee. Si las pruebas dependen unas de otras, la primera que falle provocará una sucesión de fallos, dificultará el diagnóstico y ocultará efectos posteriores.
- **Repetición:** Las pruebas deben poder repetirse en cualquier entorno. Deben poder ejecutarse en el entorno de producción, en el de calidad y en un portátil de camino a casa en un tren sin red. Sino pueden repetirse las pruebas en cualquier entorno, Siempre se tendrá una excusa de su fallo. También verá que no pueden ejecutarse las pruebas si el entorno no está disponible.
- **Validación automática:** Las pruebas deben tener un resultado booleano: o aciertan o fallan. No debe ser necesario leer un extenso archivo de registro para saber si una prueba ha acertado, ni comparar manualmente dos archivos de texto distintos para ello. Si las pruebas no se validan automáticamente, el fallo puede ser subjetivo y la ejecución de las pruebas puede requerir una extensa evaluación manual.
- **Puntualidad:** Las pruebas deben crearse en el momento preciso: antes del código de producción que hace que acierten. Si se crean las pruebas después del código de producción, puede que resulte difícil probarlo. Puede decidirse qué parte del código de producción sea demasiado difícil de probar. No se debe diseñar código de producción que no se pueda probar.[28]

Alcanzar una cobertura superior a la mencionada anteriormente significa para el equipo de la UI realizar despliegues del producto muchos más rápidos evitando la generación de documentación adicional y protocolos. Dentro de la empresa existe un grupo de personas que conforman el CAAB (*Change Advisory and Approval Board*), la cual es una junta que discute y aprueba los cambios que se vayan a desplegar a producción. Esta junta se encarga de recibir el RFC *(Request For Change*) junto con otros documentos, evaluarlos y de dar aprobación para los cambios.

Tener una cobertura superior al 65% significaría para el equipo ahorrar tiempo creando documentación pues con esta cobertura tendrían ya una preaprobación por parte del CAAB para realizar cambios en los entornos de producción.

7.2 Implementación y Ejecución de Pruebas Unitarias

En esta parte del proceso es cuando se realiza la configuración del archivo de pruebas perteneciente a cada componente, para ello es necesario hacer la inyección de los módulos, proveedores y

servicios que son usados por la aplicación para su funcionamiento, en algunos casos dichos módulos o servicios deben ser reescritos a forma de *mocks* para lograr un comportamiento aislado en la prueba. En esta sección, como se mencionó anteriormente es necesario definir los *mocks* de datos que serán consumidos por los métodos o retornados por los servicios en cada una de las pruebas que lo amerite, dichos *mocks* por lo general son sacados de inspecciones a la red mientras se hace uso del aplicativo para así, tratar de emular al máximo el comportamiento del aplicativo en un ambiente natural. También es posible utilizar *mocks* que ya hayan sido usados en pruebas anteriores, estos por lo general se encuentra dentro del proyecto en una carpeta llamada 'mock-data' la cual subdivide los archivos tipo JSON en categorías, esto para facilitar la lectura y búsqueda de archivos.

Angular-CLI al momento de generar cada componente crea 4 archivos asociados a este:

- Un archivo tipo TypeScript el cual almacenara la lógica del componente
- Un archivo tipo HTML donde se realizará la maquetación de la vista del componente
- Un archivo tipo CSS encargado del estilo
- Y por último un archivo Spect en el cual se deben escribir las pruebas unitarias correspondientes a ese componente.

Esta característica de Angular permite que: no sea necesario que las pruebas sean almacenadas en algún repositorio adicional, que estas sean usadas para verificar el código de una forma más sencilla cuando hay cambios y que sean mucho más fáciles de encontrar. Los resultados finales asociados a las pruebas realizadas se pueden apreciar en la tabla *Tabla 10. Resumen de pruebas unitarias a Cloud Defender y Cloud Insight*

7.3 Monitoreo de Cobertura Generada y Análisis Asociados

El reporte de cobertura es generado por medio de la instrucción en consola *npm run test*. El reporte muestra el porcentaje de cobertura de las pruebas con ayuda de 4 variables como se puede ver en la figura *Ilustración 30.* Variables de cobertura: las líneas de código cubiertas, los métodos probados, número de declaraciones cubiertas y los caminos que fueron probados en cada una de las pruebas. El manejo de dichas variables por separado hace más fácil la lectura de la cobertura mostrada por el reporte. El uso de este número de variables en los reportes de Jasmine se debe a la complejidad que requiere el leer la cobertura de cada una de las pruebas. En el caso de Alert Logic la unidad de medida usada para evaluar la cobertura de las pruebas es la cobertura de funciones.

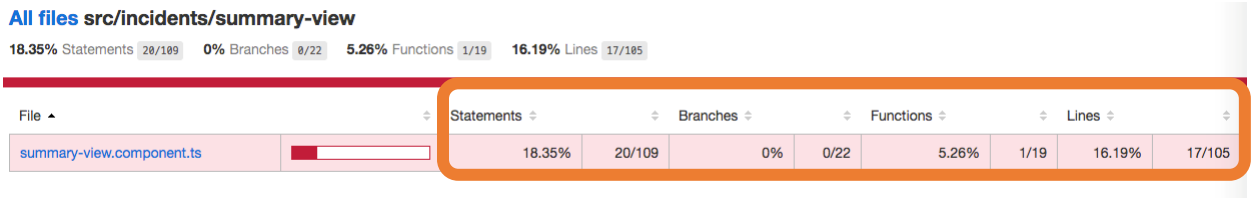


Ilustración 30. Variables de cobertura

A continuación, se presentará una composición basada en el reporte automático generado por Jasmine y los datos especificados en rally para el respectivo componente, esto con el fin de mostrar la valiosa información que nos proporcionan estas dos herramientas. Adicionalmente un resumen de los resultados de las pruebas unitarias durante el periodo de pasantía que se realizaron sobre los 18 componentes. El ejemplo de reporte consta de: a) Identificador del *sprint*, b) la historia de usuario, c) la fecha en la que fue creado el reporte, d) el desarrollador que estuvo a cargo de dichas pruebas, e) una descripción de la solicitud de pruebas, f) una solución, g) la cobertura antes de realizarse las pruebas, h) la cobertura después de terminadas las pruebas y un i.) resultado que consta de los *mocks* usados, las líneas de código que fueron consumidas por los casos de prueba y el porcentaje de cobertura final. Dicho reporte se puede observar en la *Ilustración 31. Reporte US66299*

ALERT LOGIC®
Security. Compliance. Cloud.

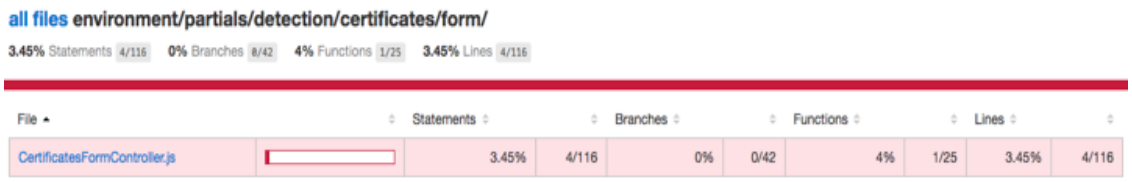
Sprint: 2017-Q4 Sprint 1 | User Story: US66299

US66299

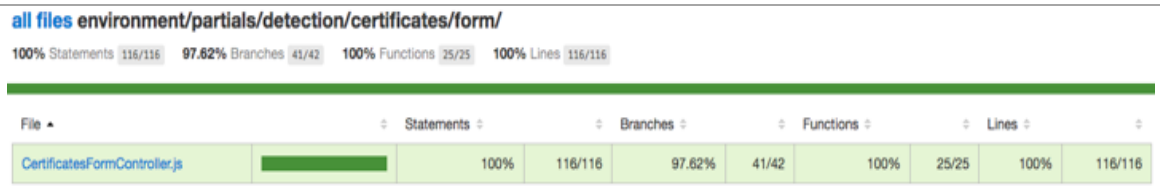
Developer: Juan Camilo Narvaez Sandoval
Date: October 12, 2017

Description:
As Alert Logic we want to increase the unit testing coverage in CertificatesFormController.js file which belongs to al-ui-threat-manager repo
Solution:
The unit tests for environment/partials/detection/certificates/form/ CertificatesFormController.test.js were created.

Before test:



After test :



Result: The coverage was increased from 3.45% to 100%, 1 mock were needed and 695 code lines to achieve the new coverage.

Ilustración 31. Reporte US66299

El reporte de los resultados de las **435 pruebas unitarias** es condensado en *Tabla 10* que contiene: a) El identificador de historia de usuario, b) el nombre del componente, c) la fecha en la que se generó el reporte, d) la cobertura inicial, e) la cobertura final, f) el número de líneas escritas para la prueba, g) el número de *mocks* usados y h) la cantidad de pruebas unitarias que se fabricaron para probar el respectivo componente, además de un consolidado final de cuantas pruebas, *mocks* y líneas de código que se realizaron durante la pasantía.

Tabla 10. Resumen de pruebas unitarias a Cloud Defender y Cloud Insight

US	Component	Date	Init coverage	Final coverage	Code lines	Mocks	Unit test
US68387	incidents-accordion-card	Oct 30, 2017	53.57%	100%	135	0	8
US68251	evidence-list-aggregate-entry	Oct 31, 2017	30%	100%	251	0	9
US68661	evidence.service	Nov 10, 2017	1.73%	98.7%	595	1	28
US69560	deployment-progress	Nov 29, 2017	34.62%	100%	733	4	22
US70462	deployment-dashboard	Dec 13, 2017	15.77%	97.1%	837	1	59
US70462	deployment-scope-selection	Dec 13, 2017	10%	98.3%	1127	1	57
US70462	deployment-summary-header	Dec 13, 2017	56.41%	97.1%	328	1	16
US71254	deployment-scalable-graph	Dec 28, 2017	16.93%	100%	874	5	46
US71254	deployment-scan-tooltip	Dec 28, 2017	17.45%	100%	654	5	30
US71782	deployment-tooltip	Janu , 2018	12.39%	100%	451	1	27
US71782	informant-messages	Janu , 2018	47.22%	99.07%	478	1	25
US71782	strawboss-tasks	Janu , 2018	42.06%	93.65%	404	1	18
US76068	aims-users	March 22,2018	4.9%	100%	1445	3	90
US73709	welcome	March 23,2018	13.9%	100%	85	0	7
US73709	account-info	March 23,2018	7.45%	100%	161	0	8
US73709	downloads	March 23,2018	4.95%	100%	371	2	24
US73709	home	March 23,2018	4.79%	100%	119	0	7
US73709	install-and-downloads	March 23,2018	15.45%	100%	109	0	6
				TOTAL	8312	23	435

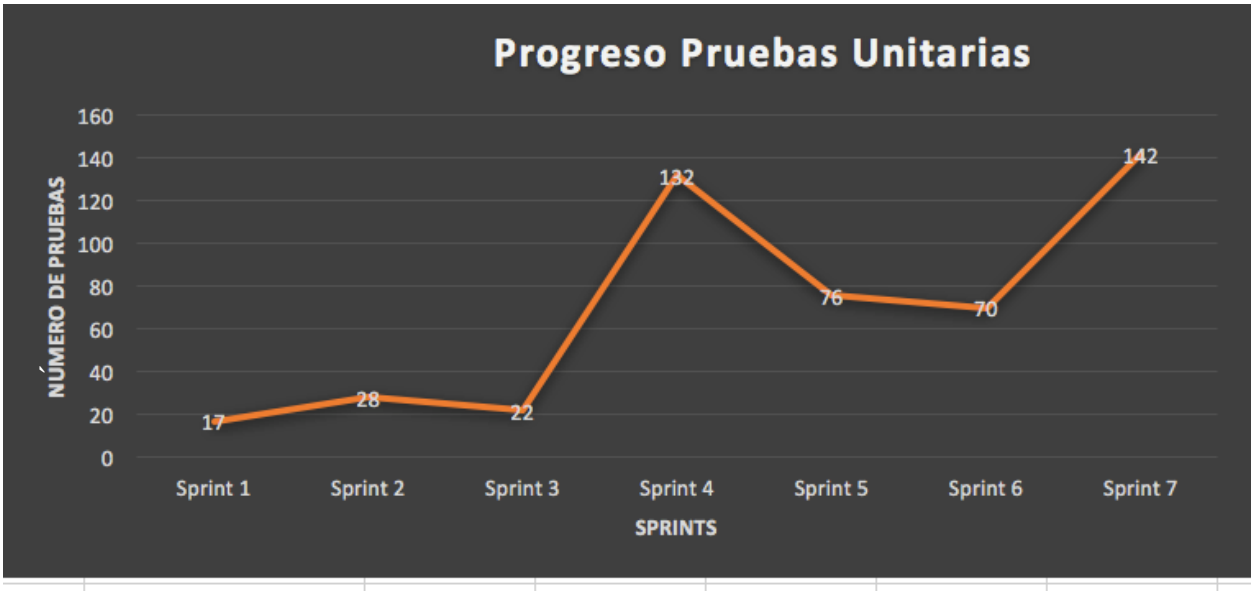


Ilustración 32. Progreso pruebas unitarias

Es necesario destacar que Alert Logic al ser una empresa que hace uso de metodologías ágiles como se ha mencionado anteriormente, el reporte final es una presentación en *Power Point* que es anexada a la historia de usuario en Rally y que consta de 2 capturas de pantalla del reporte generado por Jasmine, antes y después de la construcción del archivo de pruebas.

Ilustración 33. Resumen Objetivo #2

Uno de los principales retos que se presentó al momento de cumplir este objetivo fue el comprender la complejidad de algunos de los componentes y funcionalidades de Angular a la hora de probar, esto debido a que varias de las funcionalidades no fueron tratadas durante el periodo de

entrenamiento en pruebas, llegando a retrasar en cierta medida algunas de las entregas. Aun así y con cada uno de los improvistos, los objetivos planteados en cada uno de los *sprint* lograron ser terminados con la satisfacción del reto superado, más aún, durante el periodo en el que se nos permitió a los *interns* elegir las historias de usuario en las que deseábamos trabajar y donde siempre fue mi prioridad elegir aquellas que representaran una mayor dificultad.

Otra de las dificultades que fue visible durante este primer periodo fue la estimación de puntos para cada una de las historias de usuario, la necesidad de demostrar la capacidad de programar de cada uno de los participantes y la inexperiencia generalizada del grupo en pruebas se combinó para lograr en primera instancia una pésima estimación que obligó al grupo a realizar un esfuerzo extra que permitiera cumplir con la meta del primer *sprint*. No tener en cuenta el tiempo que toma el *code review* junto con sus respectivas correcciones, no observar detenidamente el código, y estimar con la idea de que no se presentarán obstáculos en las pruebas, fue también uno de los motivos que contribuyo a una mala estimación durante los primeros periodos.

Aunque fueron varios los obstaculos que se presentaron durante esta etapa, también es necesario resaltar que los conocimientos adquiridos en el curso profesional de 'Técnicas de pruebas de software', fueron de gran ayuda a la hora de superar muchos de estos retos planteados, ya que dichos conocimientos me permitieron guiar a mis compañeros en varias situaciones del proceso de pruebas que eran en cierta medida, desconocidas para cada uno de ellos.

Con el tiempo la estimación de puntos para las historias de usuario mejoró y se logró concluir que una buena forma de aproximar el tiempo que tardara la realización de cada prueba se podía determinar mirando el número de líneas de código del componente a probar y multiplicar ese número por 3, adicional a esto, si en los métodos se define alguna instrucción especial de la que no se tenga experiencia en pruebas anteriormente realizadas, se debe agregar ½ o 1 punto más a la estimación. Los condicionales, los *Switch* y ciclos *For* también deben ser tratados de forma especial y se debe agregar al conteo final de líneas de código de la prueba, las líneas pertenecientes al código que contiene la instrucción y luego dividir este total entre 200, para así lograr un valor aproximado.

$$Puntos\ US\ de\ Pruebas = \frac{lc + slc}{200} + 1 * cd$$

$lc = Lineas\ de\ codigo\ del\ componente$
$scl = Lineas\ de\ codigo\ metodos\ que\ contengan\ condicionales, switch\ o\ ciclos\ for$
$cd = Se\ representa\ por\ medio\ de\ 1\ punto\ por\ cada\ seccion\ de\ Codigo\ desconocido$

En conclusión, en un periodo de **5 meses** realicé **8312 líneas de código**, **23 mocks**, **435 pruebas unitarias**, en promedio 87 pruebas por mes. Esto muestra que se logró el objetivo propuesto y se superó con más 415 pruebas unitarias adicionales. Con esto se logró alcanzar el porcentaje que permite la preaprobación de la aplicación de *configurations*.

8. Corrección de Defectos a Funcionalidades de Cloud Defender y Cloud Insight

El mantenimiento o mejora de un software desplegado con problemas, puede requerir más tiempo que el desarrollo inicial del software. Cuando de solución de defectos se trata, es posible que se tenga que incorporar o modificar código que no se ajusta al diseño original con el objetivo de solucionar un problema, añadir funciones faltantes o ampliar la funcionalidad para un cliente.

A continuación, en esta sección será descrito el proceso que utiliza el equipo de la UI Alert Logic para la solución de defectos. Para posteriormente pasar a describir cada uno de los defectos que fueron resueltos.

8.1 Proceso de Reporte de Defectos

Alert Logic es una empresa que está conformada por muchos equipos de desarrollo que se encargan de distintas partes de los productos. Cuando un defecto es reportado, el primer filtro por el que debe pasara es hablar con miembros del servicio al cliente, el cual es conformado por personal experto en los productos de Alert Logic. Estas personas verifican el caso del cliente y tratan de darle una solución. En algunas ocasiones la solución se puede proporcionar inmediatamente, sin embargo, en algunos casos es necesario el arreglo de algún defecto en el producto. Cuando esto sucede los casos se envían a revisión para identificar a qué equipo le corresponde solucionar el defecto, tema que se decide por medio de reuniones con el VTT (*Virtual Triage Team*).

En caso de que el defecto corresponda a un error en la UI, este debe ser analizado, se debe proponer una fecha de entrega y una historia de usuario que debe ser agregada al *product backlog* del equipo en Rally. La fecha de entrega de la solución a los defectos se calcula a partir de una tabla de valores *Ilustración 34. Modelo de las 9 cajas*, en la cual se tiene en cuenta el impacto que tiene el defecto para el cliente, el tipo de cliente y la urgencia de solucionar este defecto. Esta tabla tiene en cuenta el tipo de cliente que se está viendo afectado, ya que para Alert Logic es de mayor importancia dar solución a un defecto que afecta a un cliente que usa una mayor cantidad de servicios de la empresa y que por ende paga una gran suma de dinero, que un cliente que usa un poco cantidad de servicios y que no genera muchos ingresos a la empresa. Dependiendo de la fecha de entrega se priorizan los casos de soporte.

URGENCIA	Normal	Alto	Urgente
urgente	Corrección de un problema o un trabajo insatisfactorio. **Segmento de clientes:** Pequeños y medianos **ETA provisto:** 1 semana **Tiempo de inicio:** Semana siguiente **SLO:**75 Días. **5**	Producto con una solución no satisfactoria, La solución puede representar una disminución en el nivel del servicio. **Segmento de clientes:** Medianos y grandes **ETA provisto:** 24 Horas **Tiempo de inicio:** Sprint Siguiente **SLO:** 37 Días **7**	El producto es impactado significativamente para un único cliente a causa de un problema en el software. El cliente no está recibiendo el servicio. **Segmento de clientes:** Todos **ETA provisto:** 24 Horas **Tiempo de inicio:** ASAP/ Sprint Siguiente SLO:1-7 Días **9**
Normal	Corrección de un problema con una solución no satisfactoria. **Segmento de clientes:** Pequeños Segmento MRR **ETA provisto:** 1 Semana **Tiempo de inicio:** Mes seleccionado para desarrollo **SLO:**105 Días **2**	Restricción de cierre de oferta para la resolución de problemas para aceptar el servicio. **Segmento de clientes:** Medianos y Grandes Segmento MRR **ETA provisto:** 1 Semana **Tiempo de inicio:** Siguiente mes **SLO:**75 Días **6**	Producto con problemas sin una solución disponible, el cliente está experimentando un bajo nivel de servicio. **Segmento de clientes:** Medianos y Grandes Segmento MRR **ETA provisto:** 24 Horas **Tiempo de inicio:** Sprint actual **SLO:**23 Días **8**
Normal	Corrección de un problema sin una solución. **Segmento de clientes:** Pequeños Segmento MRR **ETA provisto:** 1 Mes **Tiempo de inicio:** Mes seleccionado para desarrollo **SLO:**105 Días **1**	Restricción en cierre de acuerdos. Sin problemas reportados. **Segmento de clientes:** Pequeños y Medianos Segmento MRR **ETA provisto:** 1 semana **Tiempo de inicio:** mes siguiente **SLO:**105 Días **3**	Restricción en cierre de acuerdos o comportamiento de no satisfactorio de un producto con una solución planteada. **Segmento de clientes:** Medianos y Grandes Segmento MRR **ETA provisto:** 1 semana **Tiempo de inicio:** mes siguiente **SLO:** 75 Días **4**

IMPACTO AL CLIENTE

SEVERIDAD 2	**Nivel de visibilidad:** VP **Nivel de comunicación:** Diario **Proceso de Triaje:** Triaje inmediato y estatus **Status:** Basado en ETA
SEVERIDAD 3	**Nivel de visibilidad:** Director / Manager **Nivel de comunicación:** Caso de notas **Proceso de Triaje:** Proceso normal de triaje **Status:** Basado en ETA
SEVERIDAD 4	**Nivel de visibilidad:** Manager **Nivel de comunicación:** Caso de notas **Proceso de Triaje:** Proceso normal de triaje **Status:** trimestral

Ilustración 34. Modelo de las 9 cajas

Si el caso es de severidad 2, quiere decir que es de prioridad urgente, lo que significa que es necesario que sea solucionado cuanto antes. Para los casos de severidad 3 o 4 y de fácil resolución, estos deben ser marcados como casos abiertos y pasar al proceso de asignación de defectos anteriormente mencionado; en caso de no ser necesaria una la solución por medio de implementación de código, este defecto debe pasar a ser cerrado.

En caso de no ser un defecto, sino un requerimiento, Product Management evalúa la solicitud de funciones en función de su proceso y se encarga de la comunicación con los equipos y clientes. Cuando el defecto es solucionado, este pasa a ser revisado por alguno o algunos de los integrantes del equipo el cual luego de ser aceptado es mezclado en integración y el defecto pasa a ser cerrado y liberado.

Ilustración 35. Proceso de servicio al cliente en Alert Logic

Los defectos vienen con una descripción general y una lista de pasos que deben ser seguidos para replicar dicho evento. Luego de la simulación del defecto se debe crear una rama en el repositorio local donde se realizarán los cambios para corregir el error, el cual posteriormente pasara a ser revisado por uno o más desarrolladores que se aseguraran de que el defecto fue solucionado correctamente. En la *Ilustración 36* se puede observar el proceso de solución de defectos que es desarrollado por el programador al que fue asignado el defecto.

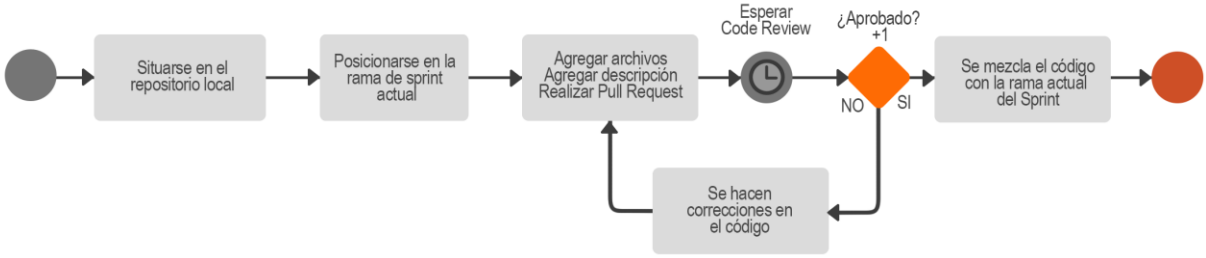


Ilustración 36. Proceso para dar solución a un defecto

8.2 Corrección a Defectos Reportados por Clientes

Para poder comprender el proceso de corrección de defectos es necesario entender de una manera general el entorno de desarrollo de Cloud Defender, en este cada desarrollador posee un nodo en la nube el cual es un entorno de desarrollo completo con características similares a los entornos de producción del Cloud Defender en cuanto a la configuración mas no en escala de este. Los nodos son básicamente máquinas virtuales que deben ser accedidas por medio de una VPN y haciendo

uso del protocolo SSH. Estos son generados por un sistema llamado el EMM (*Environment Manager*) es el encargado de realizar este proceso al crear la máquina virtual junto con todos los programas y repositorios necesarios para trabajar.

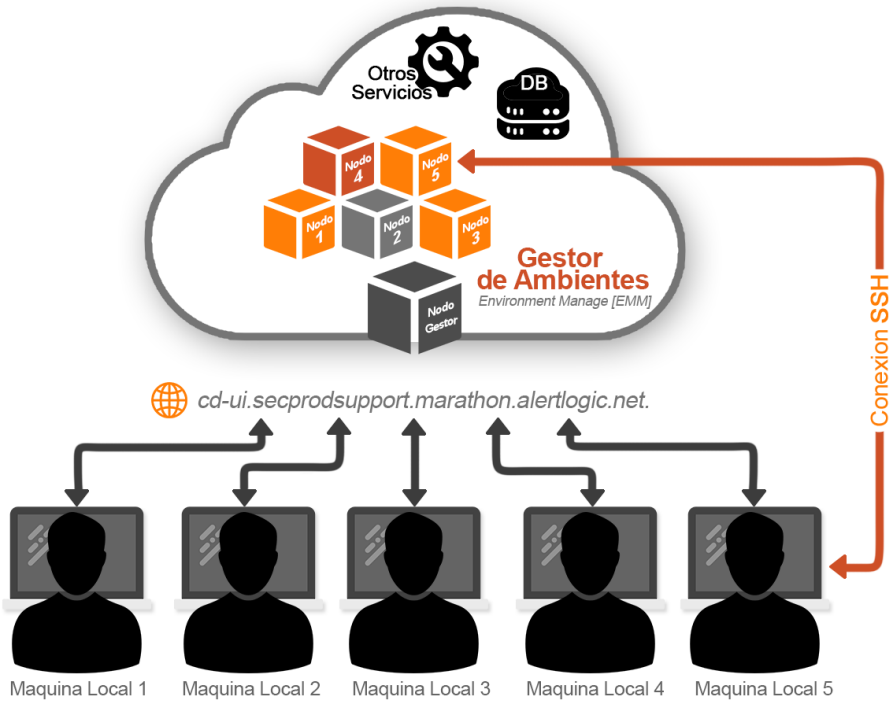


Ilustración 37. Esquema de Nodos

Todos los nodos que son usados para desarrollo comparten la misma base de datos y a otros servicios comunes como la grilla, sin embargo, en cada nodo la vista web es independiente, lo que permite a los desarrolladores realizar cambios en la lógica e interfaz sin afectar la vista respectiva de cada uno de sus compañeros.

Para realizar cambios en el nodo es necesario que el desarrollador use un editor de código capaz de conectarse al nodo para realizar los cambios necesarios como por ejemplo Visual Studio Code. Para que un desarrollador pueda ver los cambios que ha realizado en el producto se debe acceder a la página ***cd-ui.secprodsupport.marathon.alertlogic.net*** a través de una autentificación el usuario podrá ver la vista que corresponde a su nodo. Todas las peticiones que llegan a esta página son recibidas por un nodo gestor que actúa como un balanceador de carga, y redirige las peticiones al nodo que corresponde a la cuenta que se encuentra autentificada en el sistema.

A continuación, se describirán los 2 defectos reportados por clientes a los que se les dio su respectiva solución durante la pasantía. Cada uno de ellos está acompañado de su código de identificación, la descripción del defecto, dos capturas de pantalla: una captura de pantalla que muestra el error y una que muestra la solución del defecto.

DE8049 → Descripción inválida en aplicación Scans.

Problema: El defecto fue descrito por uno de los clientes como, *"En la página para 'Programar una exploración de PCI' en la interfaz de usuario indicamos que 'Los nombres de host no deben tener rutas URL adjuntas'. Esto es incorrecto. Los nombres de host con las rutas URL son compatibles ahora. La orientación informativa en la UI es correcta. En este caso, debería decir algo como 'Los nombres de host pueden tener rutas de URL incluidas'."*

Solución: Este defecto tardó 1 día en solucionarse, y tan solo fue necesaria la edición de una cadena de texto en dos archivos almacenados en los nodos.

La interfaz de usuario antes de resolverse el defecto se veía de esta forma:

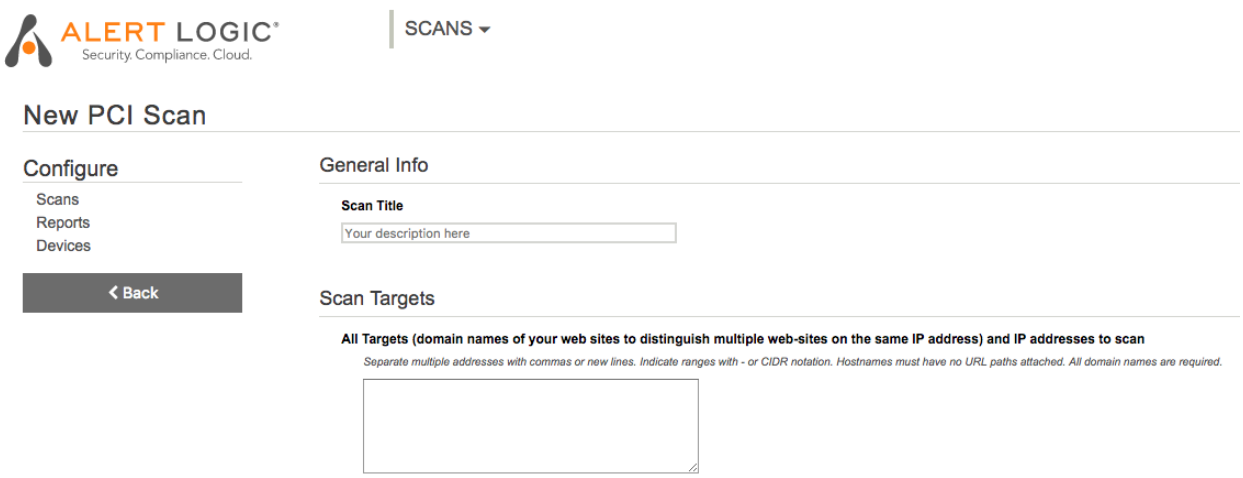


Ilustración 38. Defecto Antes

Después de resolver el defecto, la interfaz de usuario quedó de la siguiente manera:

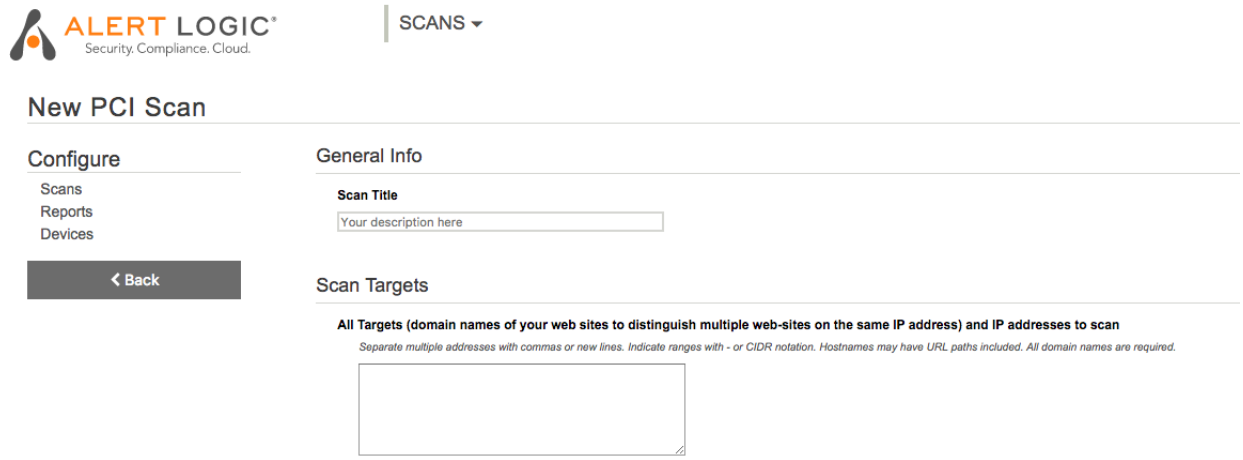


Ilustración 39. Defecto después de ser solucionado

DE7952 → Problemas en el almacenado de una Política de correlación con valor 0

Problema: El defecto fue descrito de la siguiente manera, *"Cuando se crea o edita una Política de correlación, las condiciones que son definidas con valor 0, no son guardas correctamente en el servidor. Esto se puede evidenciar cuando se intenta acceder de nuevo a esa política, no se puede ver el cambio guardado."*

Solución: Tardó 1 mes dar solución completa a este defecto. Esto a causa de distintos motivos ya que en un principio se pensó que esto ocurria a causa de alguna de las validaciones realizadas por el Front-End desarrollado en Angular en el proceso de envío del formulario al servidor para ser almacenado, pero a medida que se fue explorando en el código se pudo encontrar que el error se encontraba del lado del nodo el cual está escrito en PHP. Este defecto fue resueltó junto a Brayan Stiven Tabarez, también integrante del programas de prácticas y pasantias Alert Logic.

La interfaz de usuario antes de resolverse el defecto se veía de esta forma:

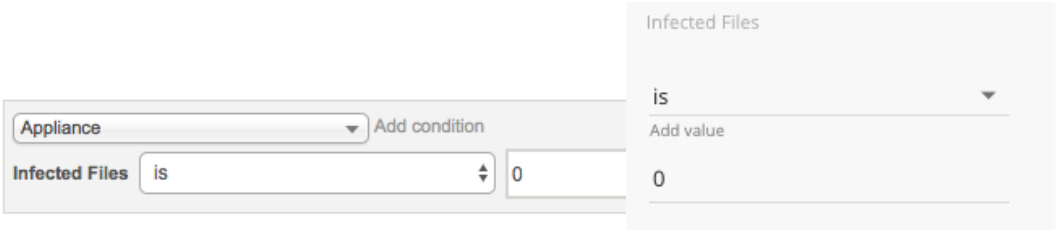


Ilustración 40. Defecto antes de ser resulto

Evidencia de solución del defecto en el JSON que contiene los datos que seran almacenados:


```
"propertyConditions": [
  {
    "id": [redacted],
    "property": "sensor_id",
    "propertyValue": null,
    "propertyValueType": null,
    "propertyValueInvert": 0,
    "propertyValueCase": false,
    "name": "Appliance"
  }
],
"tokenConditions": [
  {
    "id": 151,
    "parsedTokenId": "24B75BCF-[redacted]",
    "tokenValue": "0",
    "tokenValueType": "IS",
    "tokenValueInvert": 0,
    "tokenValueCase": false,
    "description": "Infected Files"
  }
],
"conditions": [
  {
    "rule_id": [redacted],
    "id": "24B75BCF-[redacted]",
    "value": "0",
    "type": "is",
    "invert": 0,
    "case": false,
    "desc": "Infected Files"
  }
],
```


Ilustración 41. Muestra de solución del defecto

8.3 Corrección a Defectos encontrados por Pruebas

Durante el proceso de desarrollo de software los errores pueden ser encontrados no solo después de la liberación del software a los usuarios, la pre-etapa de *testing* también es un buen momento para encontrar bugs, siendo esta su principal motivación.

Los errores que serán presentados fueron causados por: pruebas mal diseñadas, problemas con la migración de funcionalidades de un lenguaje a otro, bugs encontrados durante la creación de pruebas unitarias, y defectos reportados por los líderes. A continuación, se harán presentes los 3 defectos que fueron encontrados durante la etapa de pruebas, estos constan de una pequeña descripción, un ejemplo del error de lógica y su respectiva solución.

Defecto 1: Durante el proceso de *code review* del test para el componente *incident-detail* se encontró que en la función *loadIncidente( )* cuando define el *header.itemId*, toma el valor indefinido ya que no encuentra atributo *ItemId* gracias a que la estructura de respuesta de *utilities.getHeaderTerceryConfig(this.incident)* cuando se realiza un acceso al Array donde no se encuentra el valor esperado, este devuelve -1, lo cual es tomado por el condicional como algo verdadero, lo cual causa problemas en la lógica

Tabla 11. Lógica de la solución del defecto 1

SOLUCION DEFECTO 1 POR PRUEBAS	
DEFECTO	SOLUCION

If(array.indexOf('this.incident')){
header.itemId = array['this.incident]
}
	If(array.indexOf('this.incident')>=0){
	header.itemId=array['this.incident']
	}

Defecto 2: Este defecto se hizo presente en el momento de la ejecución de las pruebas, donde se pudo evidenciar un comportamiento extraño al momento de remover una red de un arreglo, esto dado que la función *indexOf* al no encontrar un elemento dentro del Array, este retorna -1 como respuesta, este -1 al pasar a la función *Slice()* esta lo que hace es eliminar el último elemento del arreglo, devolviendo así una respuesta incorrecta ya que en lugar de almacenarse las redes que se encuentra aún activas, este elimina la última en la lista

Tabla 12. Lógica de la solución del defecto 2

SOLUCION DEFECTO 2 POR PRUEBAS	
DEFECTO	SOLUCION

removeNet = function(item){
var itemIndex = view.policy
.monitoring
.nets
.indexOf({id: item.id});
view.policy.monitoring.nets
.splice(itemIndex, 1);
updateEmbeddedPolicy();
}
	removeNet = function(item){
	var itemIndex = view.policy
	.monitoring
	.nets
	.indexOf({id: item.id});
	if (itemIndex>=0) {
	view.policy.monitoring.nets
	.splice(itemIndex, 1);
	updateEmbeddedPolicy();
	}

Defecto 3: En este caso los múltiples defectos fueron encontrados gracias a la firma de cada método, la cual definía el retorno de un valor booleano luego de la evaluación lógica de una serie expresiones que determinan si se encuentran definidas una serie de variables que hacen referencia a las contraseñas asociadas a un usuario, es necesario comprender que esta es una validación de campos a la hora de restablecer o cambiar una contraseña, y no está vinculado al proceso de autentificación de la aplicación. Este problema de respuesta en los métodos se ocasionó al momento de migrar el código de PHP a TypeScript ya que las sentencias lógicas encerradas en paréntesis tienen un comportamiento diferente pero una estructura igual al ser definidas.

```
/**
 * Return true if any including old password field has data.
 * @return {boolean} true if password field has data.
 */
anyPasswordFieldHasData = () => {
    return (this.txtPasswordOld || this.txtNewPassword || this.txtPasswordConfirm);
};

/**
 * Return true if the password and the confirm password has data.
 * @return {boolean} true if password has data.
 */
bothPasswordFieldsHasData = () => {
    return (this.txtNewPassword && this.txtPasswordConfirm);
};

/**
 * Return true if old password match the new password
 * @return {boolean} true if the old password match the new password
 */
oldPasswordMatchNewPassword = () => {
    return (this.txtNewPassword === this.txtPasswordOld && (this.txtNewPassword && this.txtPasswordOld));
};

/**
 * Return true if the passwords fields not match
 * @return {boolean} true if password not match
 */
thePasswordsNotMatch = () => {
    return (this.txtNewPassword !== this.txtPasswordConfirm);
};
```

Ilustración 42. Lógica del defecto 2 encontrado por pruebas

Para solucionar este defecto fue únicamente necesario añadir al final de cada sentencia la instrucción condicional de una sola línea *" ? true: false;"* logrando así redefinir el comportamiento adecuado de cada uno de los métodos.

```
/**
 * Return true if any including old password field has data.
 * @return {boolean} true if password field has data.
 */
anyPasswordFieldHasData = () => {
    return (this.txtPasswordOld || this.txtNewPassword || this.txtPasswordConfirm)? true : false ;
};

/**
 * Return true if the password and the confirm password has data.
 * @return {boolean} true if password has data.
 */
bothPasswordFieldsHasData = () => {
    return (this.txtNewPassword && this.txtPasswordConfirm)? true : false ;
};

/**
 * Return true if old password match the new password
 * @return {boolean} true if the old password match the new password
 */
oldPasswordMatchNewPassword = () => {
    return (this.txtNewPassword === this.txtPasswordOld && (this.txtNewPassword && this.txtPasswordOld) )? true : false ;
};

/**
 * Return true if the passwords fields not match
 * @return {boolean} true if password not match
 */
thePasswordsNotMatch = () => {
    return (this.txtNewPassword !== this.txtPasswordConfirm)? true : false ;
};
```

Ilustración 43. Solución al defecto 2

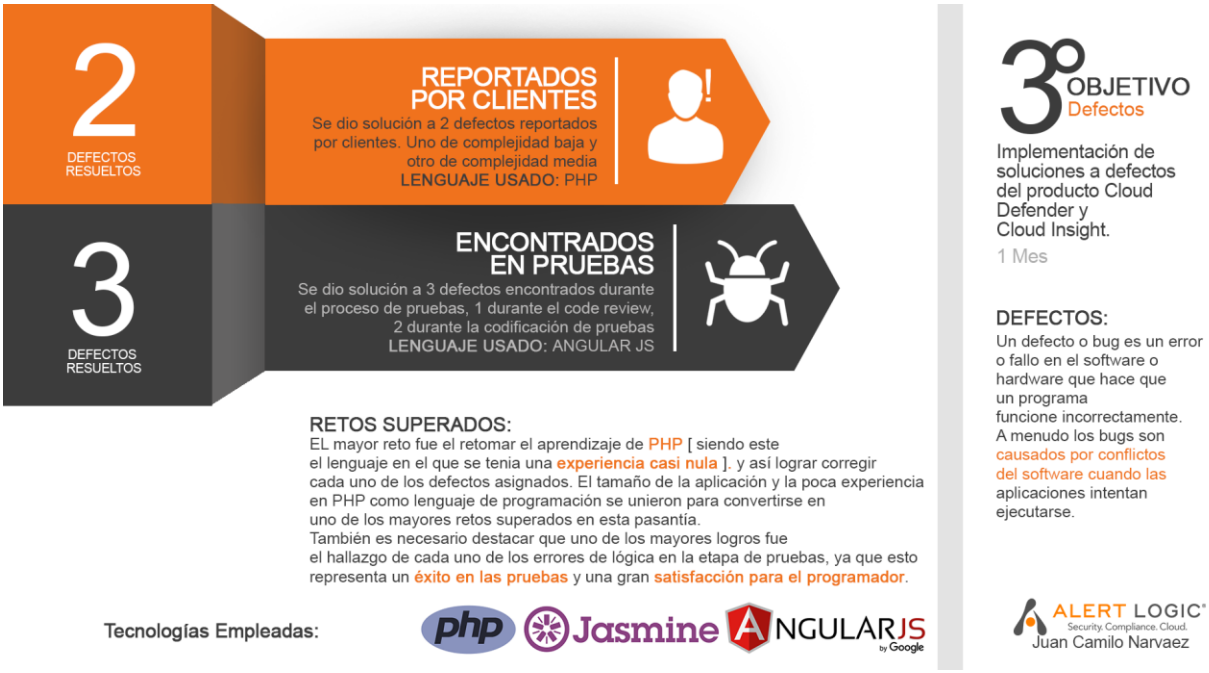


Ilustración 44. Resumen Objetivo #3

Durante este periodo, aunque de todos los objetivos fue el más corto, tuve que enfrentar uno de los mayores retos al tener que retomar el aprendizaje en PHP y así lograr corregir cada uno de los defectos asignados. PHP es un lenguaje de programación dinámico y complejo que, a diferencia de la mayoría de los lenguajes de programación más populares hoy en día como Python o JavaScript, no fue creado en el contexto de una empresa, ni un grupo de trabajo con el fin de concebir un lenguaje de programación de categoría global. Uno de los grandes reclamos que tienen los programadores con respecto a PHP es que una buena parte del código existente es código espagueti que está hecho un caos difícil de mantener, esto, junto con las múltiples posibilidades que este lenguaje ofrece, hacen que muchas veces para terceros ajenos a la construcción del código inicial, sea difícil el comprender rápidamente la lógica del código, lo cual no es una excepción para este caso. El tamaño de la aplicación y la poca experiencia en este lenguaje de programación se unieron para convertirse en uno de los mayores retos superados en esta pasantía. Aun así, es necesario resaltar que se disfrutó en gran medida el hallazgo de cada uno de los errores de lógica en la etapa de pruebas, pues eso representa un éxito en las pruebas y una gran satisfacción para el programador.

En total se corrigieron **2 defectos reportados por los clientes** los cuales pertenecen a código PHP y **3 defectos encontrados durante el proceso de pruebas** para componentes en AngularJS, logrando así el objetivo de trabajar en defectos con un total de **5 defectos reparados**.

9. Implementación de Nuevas Funcionalidades
Cloud Defender • Cloud Insight

El desarrollo de software es un proceso complejo que consta de una serie de etapas; donde la implementación de funcionalidades es una de las fases que culmina el periodo de revisión y evaluación de requerimientos.

En Alert Logic los nuevos desarrollos o cambios en la UI se hacen con base a un entregable *(deliverable)* el cual es un documento interactivo en el cual se visualizan cuáles son los cambios o nuevas secciones o páginas en la interfaz que deben ser agregados. Este archivo contiene el aspecto que debe tener el componente a desarrollar o modificar, adicional a esto, este también simula el comportamiento del componente, como por ejemplo los accesos a otros paneles desde los botones y los accesos a enlaces externos. Adicionalmente a esto, este documento también cuenta con una sección de comentarios en la cual se anexan instrucciones especiales o los enlaces externos exactos que se deben usar en la aplicación.

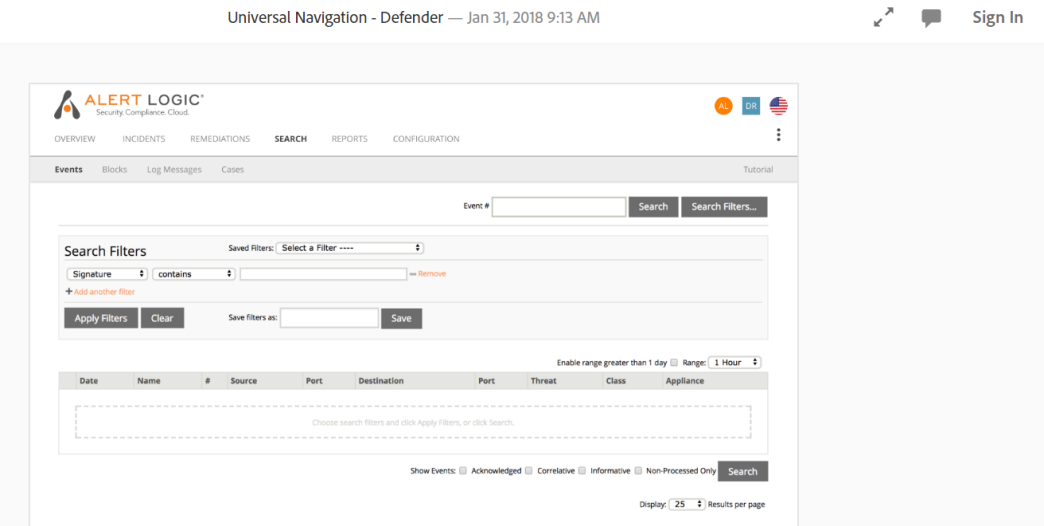


Ilustración 45. Captura de pantalla ejemplo de entregable

A continuación, se presentarán los 4 componentes que fueron desarrollados principalmente en la aplicación *O3 Portero*, todos y cada uno de ellos con la premisa general de servir de una u otra forma como ayuda o soporte para el cliente, cada uno de ellos se encuentra acompañado de una especificación inicial, las capturas de pantalla que evidencian la implementación de cada componente y una lista de aspectos importantes que forman parte de la implementación. Para lograr este objetivo fue necesaria la comprensión y uso de servicios de autentificación, información y descarga. Además de la comprensión del *framework O3* para la modelación de componentes y navegación.

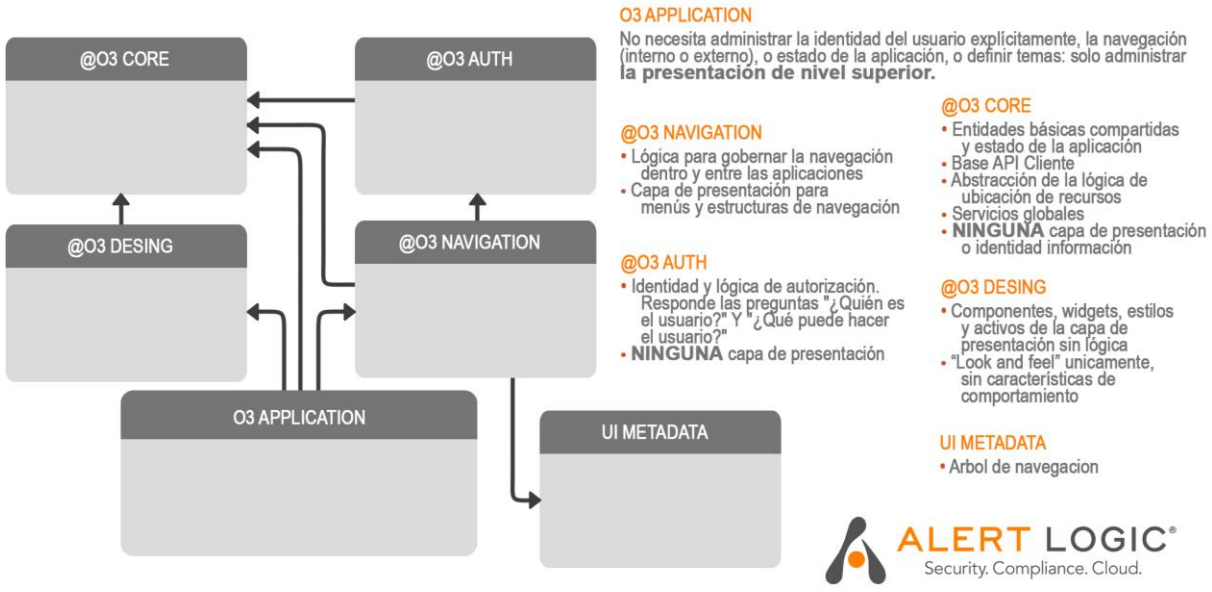


Ilustración 46. Arquitectura Framework O3

O3 es un *framework* en Angular 5 desarrollado por Kevin Nielsen, ingeniero principal de Alert Logic en los Estados Unidos. Este *framework* se encarga de separar la navegación, los servicios y procesos de autentificación de los clientes del nivel superior de presentación de cada sub-aplicación, facilitando así el desarrollo e integración de subcomponentes en la aplicación final.

9.1 Proyecto Platform → Support UI development

Especificaciones: Desarrollar el panel de Detalles de usuario el cual contiene el *Customer ID, Unique User ID, Emai*l y la *Unique Registration Key*. Las instrucciones para visitar el *Helper Center* y un enlace a la página de soporte por medio de un URL al final de las instrucciones y un botón en el menú secundario. Este panel debe ser accedido a través del menú principal superior derecho, dando clic en la opción *Support Information.*

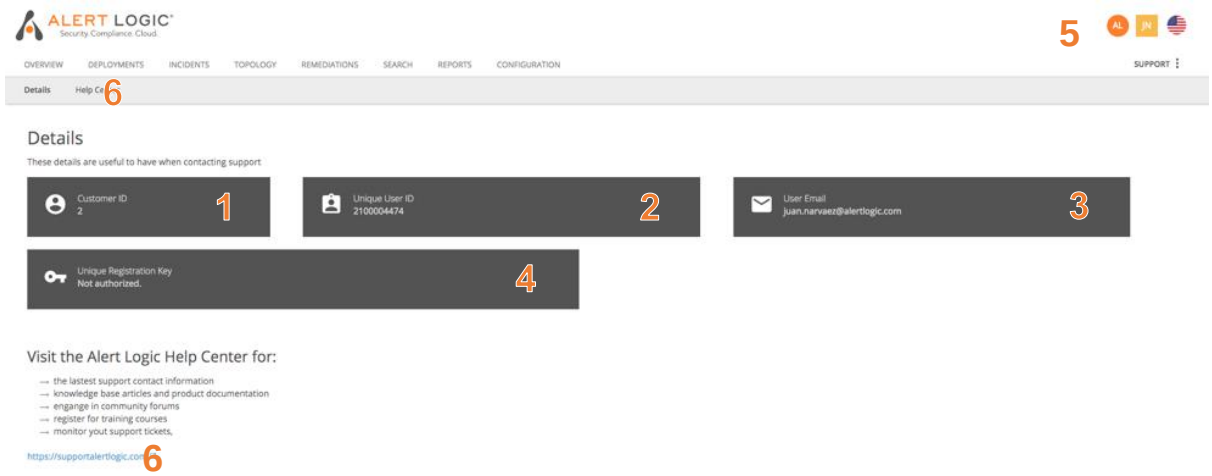


Ilustración 47. Interfaz Detalle

Descripción de funcionalidades:

1. Identificación del cliente perteneciente a la cuenta que se encuentra actualmente autentificada por medio del servicio auth y al id del cliente que es seleccionado por medio de la opción 5.
2. Identificación única de la cuenta que se encuentra actualmente autentificada.
3. Email asociado a la cuenta que se encuentra autentificada.
4. Llave de registro única asociada al ID del cliente. Esta información es obtenida por medio del servicio *this.brainstem.getStateProperty(O3AuthState.AIMSActingAccountID)*. Si el ID no tiene asociado una llave de registro única, dicha caja de información es ocultada.
5. Por medio de este botón se accede a un panel donde se encuentran todos los diferentes clientes, y donde se selecciona el cliente del que se quiere ver la información. Este botón muestra las iniciales del cliente.
6. Enlace externo a la página de soporte.

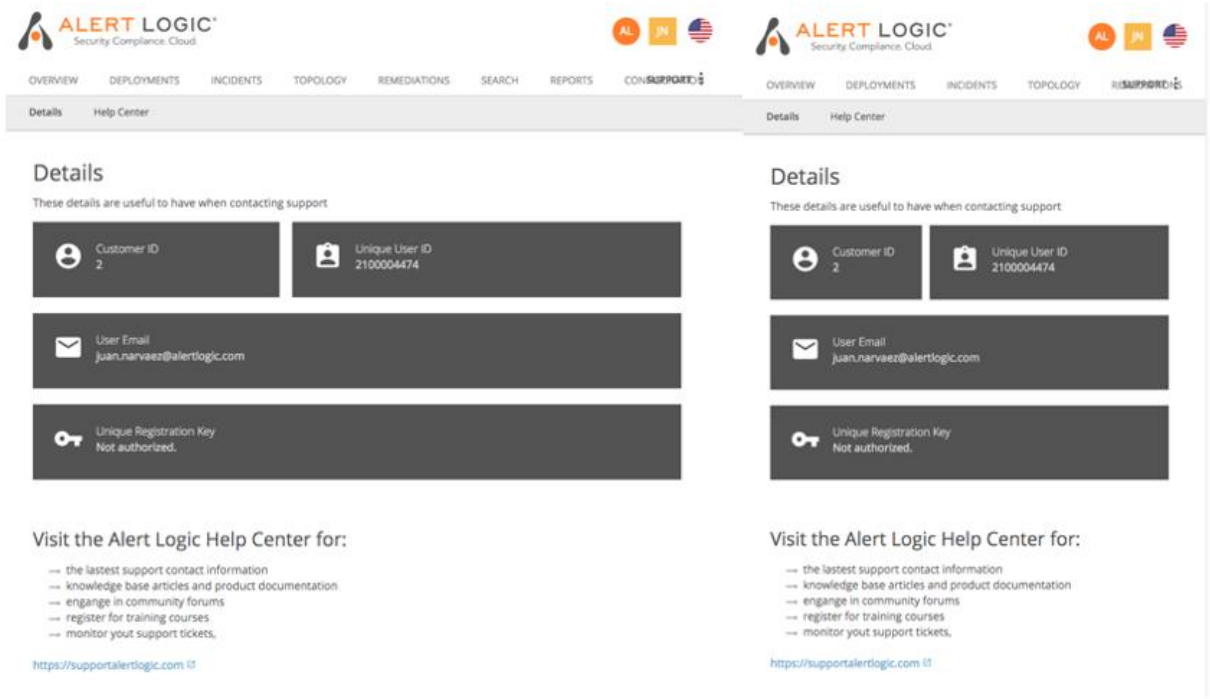


Ilustración 48. Comportamiento Responsivo Pagina de Información de Soporte

Características principales: Desarrollo completo de la plantilla CSS | Enlaces externos | Petición de información de usuario por medio de servicios | Manejo de respuestas en formato JSON | Aplicación de FlexBox Responsivo | Uso de O3

9.2 Proyecto Platform → Pagina de Guia y Descargas

Especificaciones: Desarrollar el panel de 'Install Guides & Downloads' para los productos de Threat Manager y Log Manager el cual debe contener las instrucciones para configurar su análisis de tráfico de red las cuales incluye una serie de 9 pasos en el caso de Threat Manager y una descripción más breve para el caso de Log Manager, estas instrucciones deben contener los enlaces de descarga para Linux Agents, Windows Agents, Collector y "virtual appliances" por medio de la consulta de un servicio, adicionalmente a esto, las instrucciones deben mostrarse solo y únicamente si el cliente es poseedor del producto, en caso contrario, las respectivas instrucciones no deben ser mostradas.

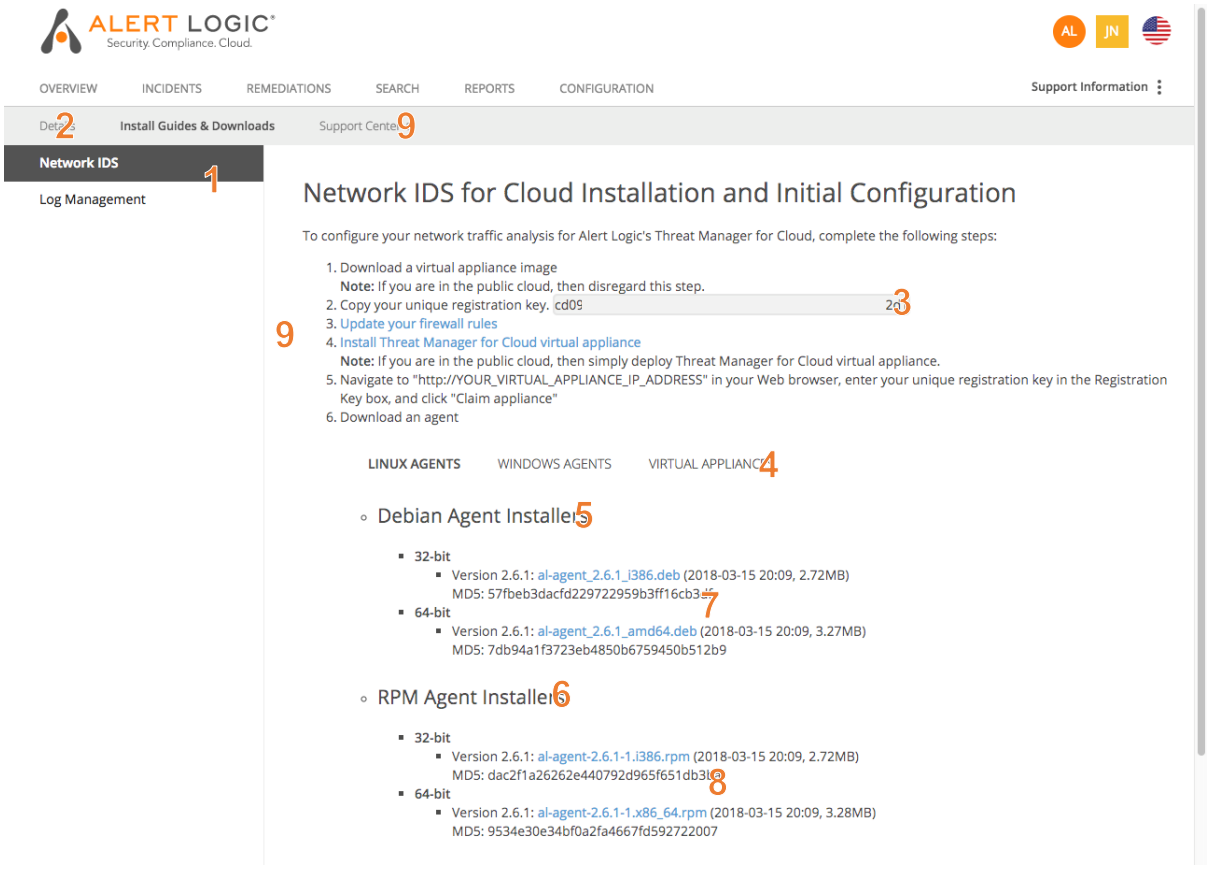


Ilustración 49. Guía de instalación para Threat Manage

Descripción de funcionalidades:

1. Menú terciario con las guías de instalación para Threat Manager y Log Manager, solo serán visibles los productos que se encuentran disponibles para dicha cuenta. En caso de ser solo uno este menú desaparece.
2. Enlace al panel de Detalle de información de la cuenta.
3. Llave de registro única, esta se encuentra asociada al ID del cliente la cual es obtenido por medio del servicio *this.brainstem.getStateProperty(O3AuthState.AIMSActingAccountID)*. En caso de no encontrarla muestra un mensaje de *"Not Found"*.
4. Sub menú de descargas para Linux, Windows y Virtual Appliances.
5. Sección de Instaladores de agentes Debian para Linux.
6. Sección de Instaladores de agentes RPM para Linux.
7. Enlaces de descarga obtenidos por medio del servicio *this.convergenceUtility.getDownloads(activeCID, this.product)*.
8. Enlaces de descarga obtenidos por medio del servicio *this.convergenceUtility.getDownloads(activeCID, this.product)*.
9. Enlaces externos.

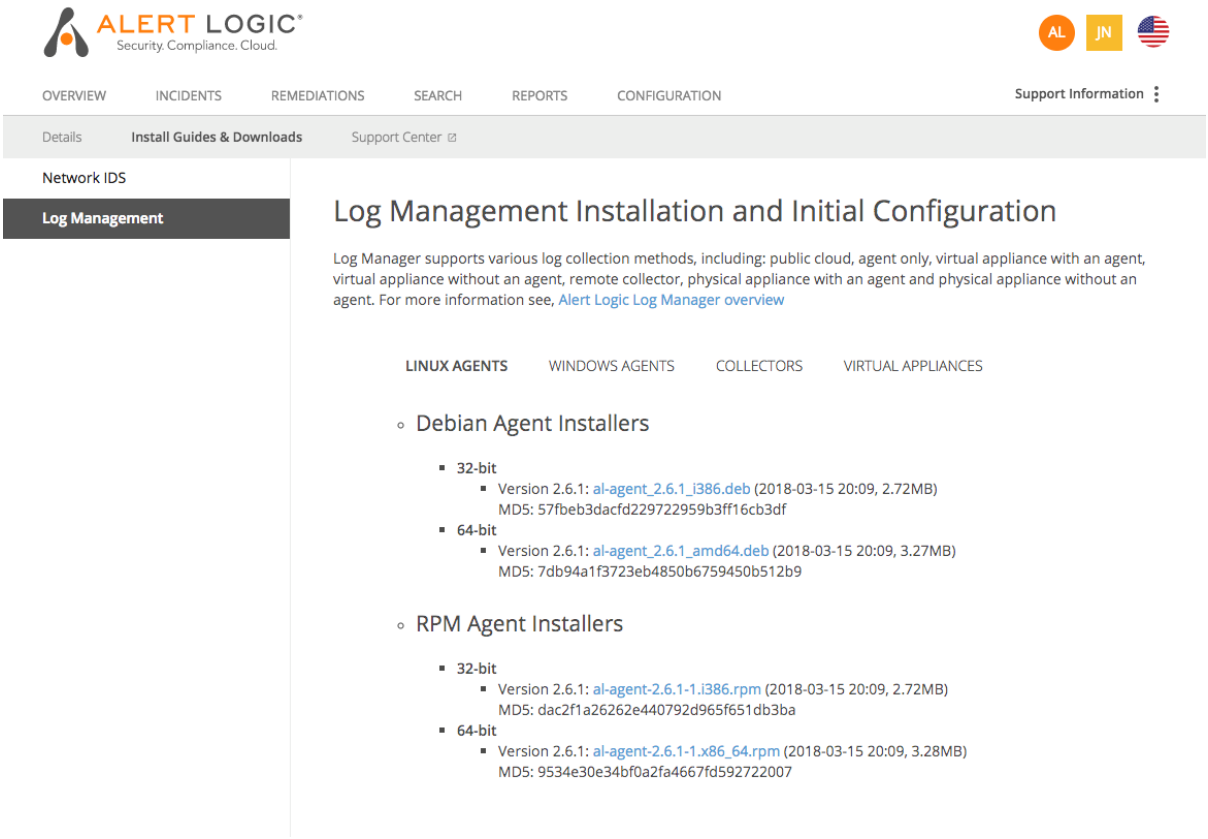


Ilustración 50. Guía de instalación para Log Manager

Características principales: Desarrollo y adaptación de la plantilla CSS | Manipulación del menú secundario y terciario por medio del framework O3 | Enlaces externos | Petición de información de usuario por medio de servicios | Petición de archivos de descarga por medio de servicios | Manejo de respuestas en formato JSON

9.3 Proyecto Platform → Pagina de Bienvenida UI desarrollada para usuarios TM

Especificaciones: Es necesario construir una página de bienvenida para los usuarios poseedores del producto Threat Manager, dicha pantalla debe contener las instrucciones finales para la configuración del producto, adicional a esto debe contener un enlace externo que permita la creación de ambientes y la instalación de artefactos y agentes. También deben especificarse los enlaces de ayuda y para servicio como parte de la cabecera de la página, de tal forma que sea visible para el usuario final y este pueda encontrarlos fácilmente en caso de necesitarlo.

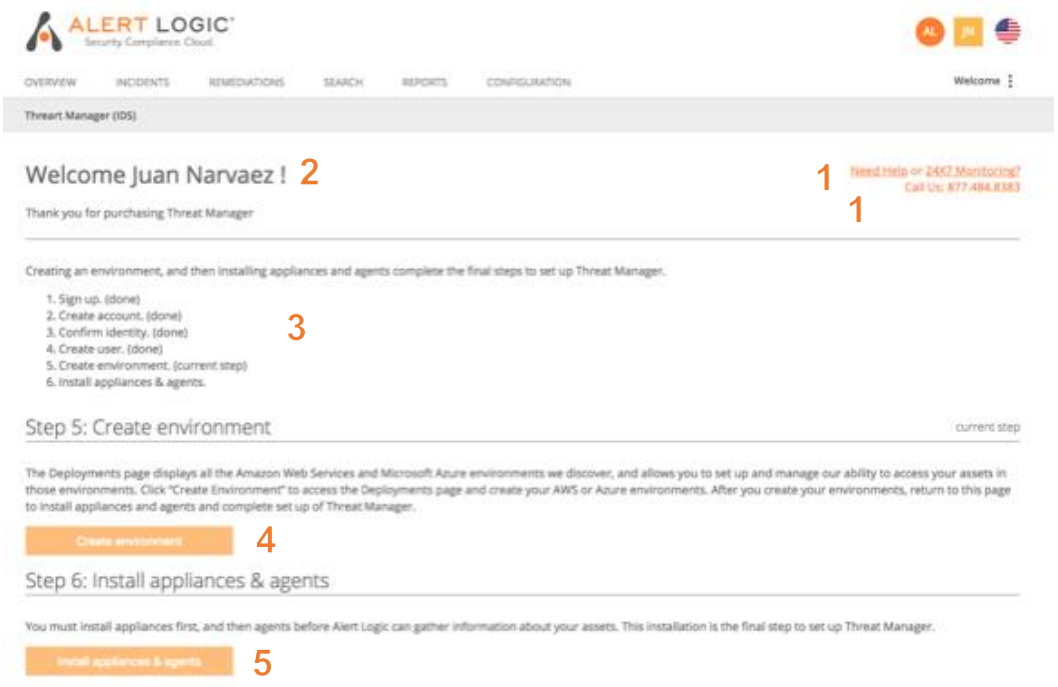


Ilustración 51. Página de Bienvenida

Descripción de funcionalidades:

1. Información de servicio al cliente, en caso de oprimir *Need Help* o *24x7 Monitoring* se abre automáticamente una ventana emergente para enviar un correo dirigido al equipo de soporte correspondiente.
2. Nombre del usuario autentificado en la aplicación.
3. Instrucciones para terminar de configurar Threat Manager.
4. Enlace externo para acceder a la página para crear un ambiente
5. Enlace externo para acceder a la página de instalación y agentes.

Características principales: Desarrollo completo de la plantilla CSS | Enlaces externos | Aplicación de FlexBox Responsivo | Uso de O3

9.4 Proyecto Platform → Documentación API Cloud Insight

Especificaciones: Es necesario adaptar la página que hace referencia a la documentación de la API de Cloud Insight al aplicativo de Accounts, dicha página debe poder ser accedida sin necesidad de estar autentificado en la aplicación. Para el caso de los usuarios que tienen cuenta y esta autentificados en la aplicación, estos podrán ver en el menú un acceso a la sección de *Deployments* que los retornara a la navegación principal de la aplicación. En esta parte de la aplicación deben especificarse 2 secciones: *Integratios* y *Api Documentation* donde la primera contiene las integraciones a las que puede ser extendido Cloud Insigth además de las API avanzadas y la

segunda la documentación API de Alert Logic que le permite automatizar algunas tareas de Cloud Insight.

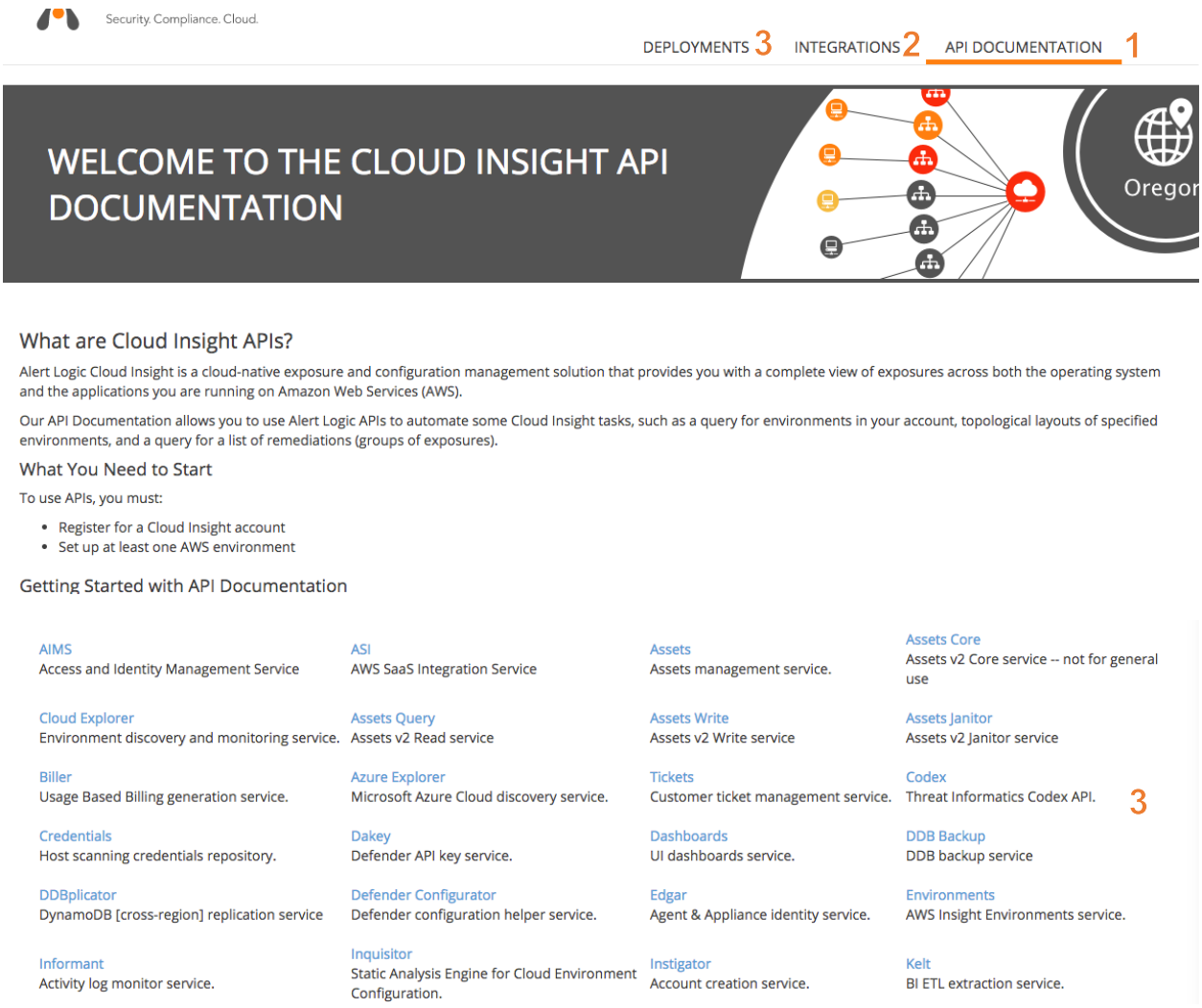


Ilustración 52. Pantalla de Documentación API

Descripción de funcionalidades:

1. Botón de acceso a la documentación de la API
2. Botón de acceso a la información de Integración
3. Botón de retorno a la aplicación, por medio de este se accede al panel de implementaciones, en caso de que el usuario se encuentre autentificado, en caso de que no, este botón estará oculto.
4. Tabla de enlaces la cual es cargada por medio de un archivo HTML que se editado por un componente externo.

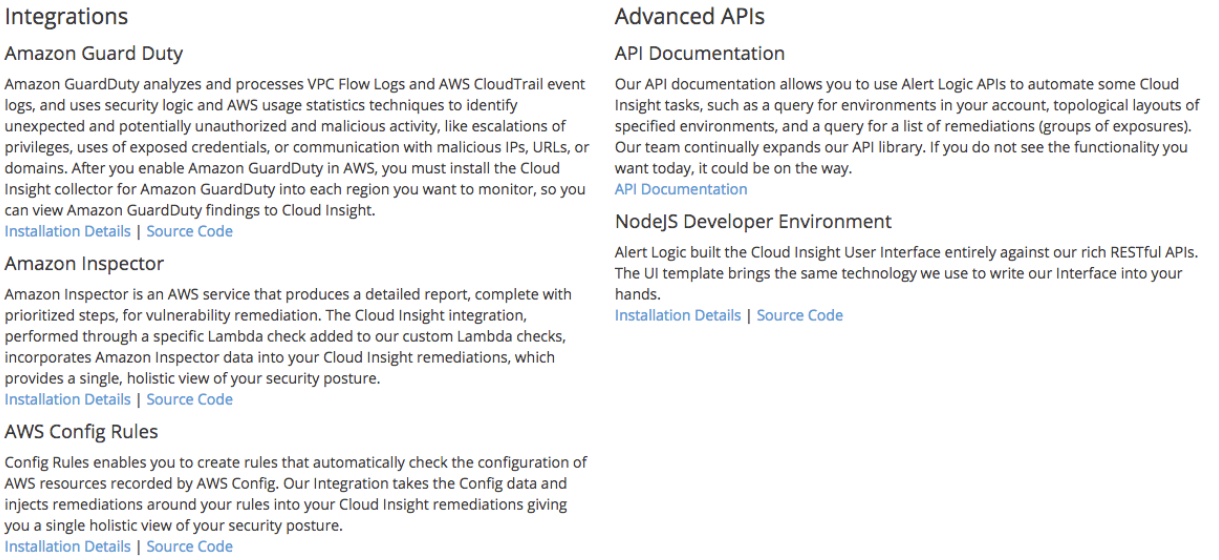


Ilustración 53. Pantalla de Integraciones

Características principales: Desarrollo completo de la plantilla CSS | Enlaces externos | Petición de información de usuario por medio de servicios | Manipulación directa en lógica O3 para modelar el comportamiento adecuado del menú en caso de usuarios autentificado y no autentificados | Aplicación de FlexBox Responsivo

Ilustración 54. Resumen Objetivo 4

El mayor obstáculo de esta etapa fue el comprender la arquitectura de las aplicaciones desarrolladas con base al *framework* propietario de Alert Logic O3, el cual tienen como característica principal el manejo de una navegación universal que es definida en un objeto tipo JSON donde se detallan los diferentes atributos que permiten a la aplicación saber cómo debe

comportarse cada menú dependiendo del componente en el que se encuentre situado, dicho *framework* facilita la unificación de componentes y la definición de menús primarios, secundarios, y terciario sin tener la necesidad de estar reescribiendo código HTML y CSS . Durante este periodo se trabajó principalmente en los repositorios *O3 Portero, O3 Navigation y UI Metadata*,

En conclusión, se logró desarrollar en un periodo de **1 mes y medio, 4 componentes** en Angular JS que constan de un total de **634 líneas de código** entre archivos tipo **TypeScript, CSS** y **HTML**.

10. Conclusiones

El trabajo de 8 meses de media jornada laboral se puede ver reflejado en la culminación exitosa de los siguientes entregables: 435 pruebas Unitarias realizadas con éxito, 23 *Mocks* a servicios para la ejecución de pruebas automáticas, 5 defectos reparados en PHP y TypeScript y el desarrollo de 4 componentes en Angular JS, los cuales se condensan en al menos 8946 líneas de código con las que se logró en primera instancia superar el porcentaje de cobertura en pruebas de la aplicación *O3 Configuration* requerido para su pre aprobación (65%), la participación en el proceso de corrección de defectos el cual pone a prueba a los desarrolladores en la difícil tarea de comprender el código de otros, más aun cuando se trata de lenguajes tan flexibles como PHP y finalmente se logró aportar activamente a la aplicación de soporte al cliente *O3 Portero* contribuyendo con la elaboración de 4 componentes.

Se aprendió y profundizo en varias tecnologías y saberes prácticos de la Ingeniería de Sistemas frente al desarrollo Web. Lenguajes como **JavaScript, TypeScript, PHP**, Marcos de trabajo como **Jasmine, Angular JS** en su versión 1, 2, 4 y 5 y el framework O3. **SCRUM** como metodología de trabajo y **Rally, Git y GitHub** como herramientas de control de versiones, fueron las tecnologías y metodologías con las que se trató en mayor medida durante esta pasantía.

Es necesario destacar que el entrenamiento brindado por Alert Logic a sus practicantes antes de integrarse a las actividades laborales cumple un papel esencial que permite al estudiante conocer y familiarizarse con los procesos más importantes de la organización, logrando así un mejor desempeño en cada uno de ellos. Aun así, es importante mencionar que, aunque el proceso de entrenamiento fue completo, el aprendizaje dentro de la empresa es continuo, por lo que cada día se asimilan enseñanzas nuevas y se enfrentan nuevos retos. Todo esto se debe en gran medida a que el cronograma de actividades propuesto por el director de la práctica de Alert Logic para el periodo de pasantía fue diseñadó de forma estratégica para que este nos permitiera adquirir de una forma cómoda, fluida y coherente los conocimientos necesarios para cumplir a cabalidad el desarrollo de cada objetivo. Por otra parte, se pudo evidenciar que la curva de aprendizaje en un proyecto de proporciones tan grandes es bastante lenta, más aún cuando gran parte de los procesos son ejecutados en un idioma diferente al nativo, en este caso el idioma inglés. Aun así, siendo un proceso que cuesta dificultad, con el tiempo, dicho esfuerzo se vio reflejado en un mejor manejo de los de los conceptos y herramientas, además de en un mejoramiento del inglés como segunda lengua.

El proceso de desarrollo de pruebas unitarias es sumamente importante, ya que gracias a este se puede lograr un mayor control sobre los cambios de estructura, arquitectura y lógica de código que

permiten y ayudan rápidamente en el hallazgo y solución de *bugs* o daños en alguna parte de la lógica de los componentes. Alert Logic utiliza las funciones como parámetro para la evaluación de cobertura en las pruebas unitarias, esto significa que el porcentaje de cobertura dependerá de cuántas funciones lógicas han sido ejecutadas por las pruebas, sin importar el flujo de datos de cada una de ellas. Este parámetro, aunque es un bueno a la hora de analizar qué tantas funciones del código se pusieron a prueba durante cada *test*, al final resulta ser un poco engañoso si se compara con otros parámetros de evaluación como lo puede ser la cobertura por ramas, la cual proporciona una mejor lectura de que partes de la lógica del código fue probada y cuáles no, calculado su porcentaje de cobertura con base a la cantidad de caminos en el flujo de datos del aplicativo probo, lo que significa que dicho porcentaje a groso modo reflejaría la cantidad de comportamientos que puede ocupar la aplicación para distintos sets de datos entrantes. Esta decisión de tomar las funciones acelera el proceso de aprobación, pero puede afectar en cierta medida la calidad del software porque los casos de prueba pueden no ser los mejores.

Alert Logic es una empresa que se encuentra en crecimiento y aunque ha realizado un excelente esfuerzo en la implementación de pruebas unitarias para cada uno de sus frentes de desarrollo, aun así, esta puede mejorar en mayor medida el proceso de integración y entrega continúa agregando pruebas de integración y pruebas funcionales a su portafolio de pruebas automatizadas. La conformación de un grupo interdisciplinario encargado de toda la fase de pruebas que permita llevar un seguimiento completo de todo el proceso podría mermar aún más el impacto de los errores que se presentan al momento de hacer las respectivas liberaciones de software.

Durante el periodo de pasantía se pudo comprobar la efectividad de la metodología ágil SCRUM a través de su ejecución en tiempo real. El uso de SCRUM permitió que la comunicación por parte del equipo fuera constante, ayudando así a aumentar la cantidad y calidad de cada una de las entregas. El trabajo en equipo y la comunicación dentro del grupo de desarrollo es el aceite que permite que todos los engranajes funcionen correctamente; una excelente comunicación entre los miembros del equipo converge en una sólida estructura de organización capaz de agilizar y afrontar procesos evitando recaer en errores del pasado.

Por otro lado, también se pudo concluir que por más perfecto que un producto de Software se vea durante la fase BETA, difícilmente este se mantendrá libre de fallos luego de su liberación a los clientes, más aún cuando de un software complejo se trata. Por tal motivo, es importante siempre considerar en el proceso de software un espacio para el soporte y reparación de defectos ya que estos pueden llegar a ser tan críticos que pueden conllevar a la pérdida de clientes. El ciclo de vida del software es un proceso complejo que está lleno de cambios y de toma de decisiones importantes, esto se pudo ver reflejado durante el proyecto en las múltiples decisiones que se tomaron sobre Cloud Insight y Cloud Defender.

Las tecnologías asociadas a los profesionales afines a los productos de software y sistemas avanzan constantemente, por lo que ser Ingeniero de Sistemas requiere estar dispuesto a vivir una vida profesional llena de aprendizaje continuo e investigación constante. Es necesario agradecer a la Universidad Del Valle como alma mater la cual me preparó académicamente para afrontar con tenacidad todos estos retos y debo destacar asignaturas como: IPOO, Desarrollo de Software I y II, Técnicas de Pruebas de Software, Aplicaciones en la Web, Vida Artificial y Sistemas Emergentes ya que gracias a los conocimientos adquiridos en estas materias facilitaron importante tarea de alcanzar cada uno de los objetivos.

Más allá del cumplimiento de cada uno los objetivos de este trabajo de grado, es necesario destacar la experiencia enriquecedora tanto profesional como social que se obtiene con el trabajo diario dentro de una empresa y un equipo que coopera constantemente para ofrecer lo mejor de sí. Esta pasantía fue una experiencia llena de retos, aprendizajes y aportes que me ayudó en gran medida a mi crecimiento personal y profesional como Ingeniero de Sistemas de la mano de una compañía de talla mundial.

11. Trabajo Futuro

Como trabajo futuro se propone desarrollar un aplicativo o script que permita automatizar la creación del esquema inicial de pruebas unitarias en Jasmine para cada componente, servicio o clase. Este es un proceso tiende a ser repetitivo en cierta medida, por lo que automatizar parte de este proceso impactaría positivamente la fase de desarrollo de pruebas evitando que el programador se preocupe por las configuraciones iniciales, mermando el tiempo de desarrollo en cada prueba y por ende ampliando significativamente el número de entregas en cada sprint.

Referencias

[1] IBM, "¿Qué es DevOps?" [Online]. Available: https://www.ibm.com/cloud-computing/es-es/learn-more/what-is-devops/. [Accessed: 01-Apr-2018].

[2] J. Pérez Porto and A. Gardey, "Red LAN," 2015.

[3] Alert Logic, "Alert Logic® – About-us." [Online]. Available: https://www.alertlogic.com/solutions/. [Accessed: 20-Jul-2017].

[4] Microsoft, "Architecture Strategies for Catching the Long Tail. - What Is Software as a Service." .

[5] L. Joyanes Aguilar, "La Computación en Nube (Cloud Computing): El nuevo paradigma tecnológico para empresas y organizaciones en la Sociedad del Conocimiento." Revista de las facultades de Derecho y ciencias Economicas y Empresariales, Madrid, 2009.

[6] P. World, "3 factores claves para elegir un servicio de almacenamiento en la nube." 2016.

[7] C. E. Molina, "Protocolo de datagramas de usuario (UDP)," 2014.

[8] N. Macias, "Arquitecturas Multi-Tenant." 2010.

[9] C. Arsenault, "HTML vs HTML5 – What's the Difference?," 2017.

[10] Mozilla, "Learn to style HTML using CSS," 2017. [Online]. Available: https://developer.mozilla.org/en-US/docs/Learn/CSS.

[11] J. J. Tortajada Cordero, "La Guía Definitiva Del Diseño Web: HTML, XHTML, CSS Y Herramientas De Diseño," 2014.

[12] D. Barcia, "¿Que es CSS?" Platzi, 2017.

[13] W3C, "El W3C de la A a la Z." [Online]. Available: https://www.w3c.es/Divulgacion/a-z/index.html.es.

[14] M. A. Alvarez, "DOM y Compatibilidad con navegadores." DesarolloWeb.com, 2008.

[15] M. del Gerrero, "Angular JS La guia definitiva."

[16] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, "design patterns elements of reusable object-oriented software."

[17] J. Rodriguez, "Watch how the apply runs a digest," 2013.

[18] Oracle, "Mysql Description general."

[19] W3C, "SOAP Version 1.2 Part 1: Messaging Framework (Second Edition)."

[20] F. J. Lopez-Pellicer, R. Béjar, M. A. Latre, J. Nogueras-Iso, and F. J. Zarazaga-Soria, "GitHub como herramienta docente." Universidad de Zaragoza, 2015.

[21] P. Labs, "https://jasmine.github.io/2.0/introduction.html," 2017.

[22] I. Sommerville, *Ingenieria de Software*, 9th ed. 2011.

[23] P. A. ORG, "Qué es SCRUM."

[24] M. T. Gallego, "'Metodología Scrum.' Gestión de Proyectos Informaticos." 2012.

[25] Alert Logic, "Alert Logic® – Solutions." .

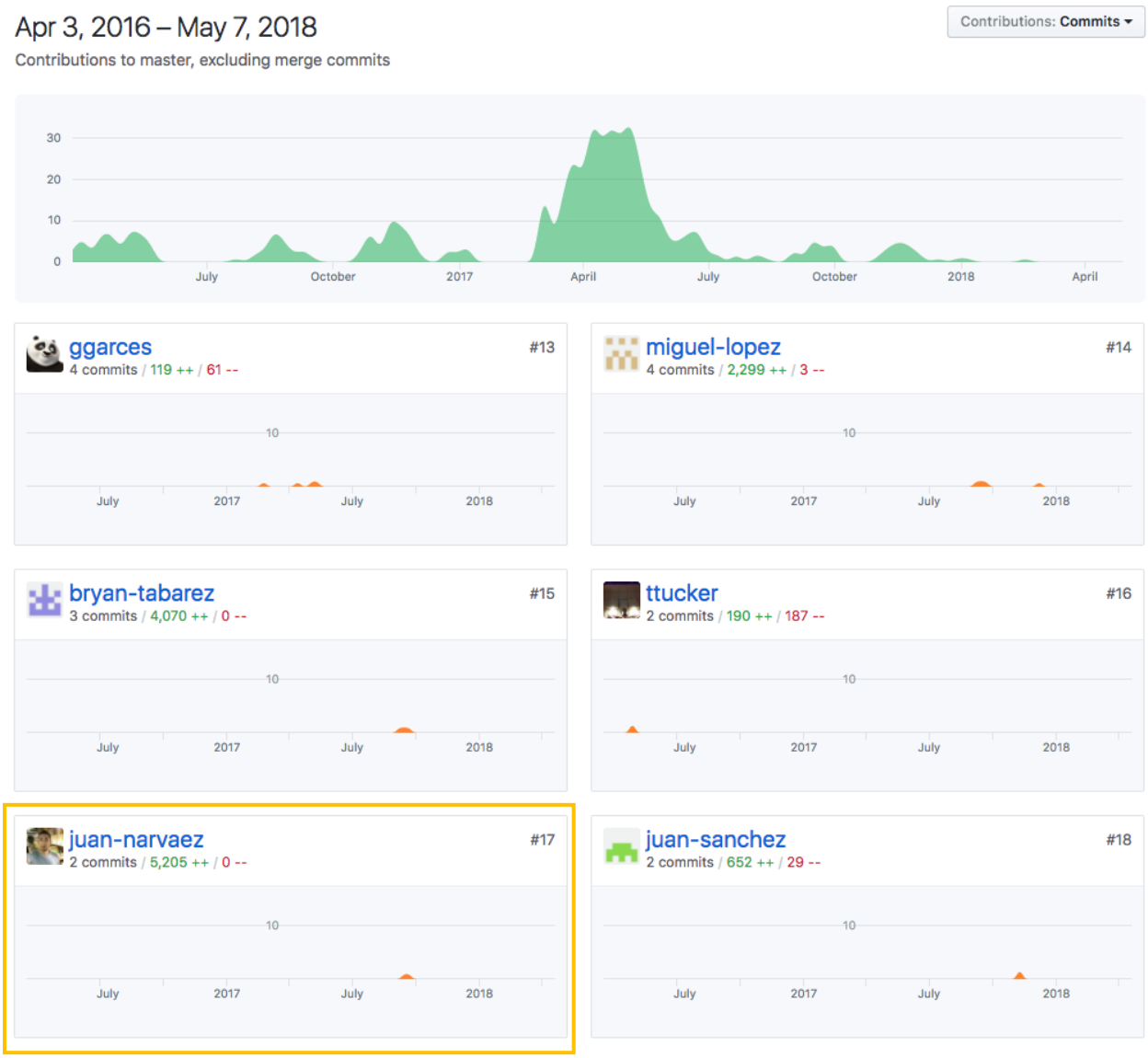
[26] Alert Logic, "How it works - Cloud Insight." .

[27] TMDB, "The Movie DataBase." [Online]. Available: https://www.themoviedb.org/?language=es.

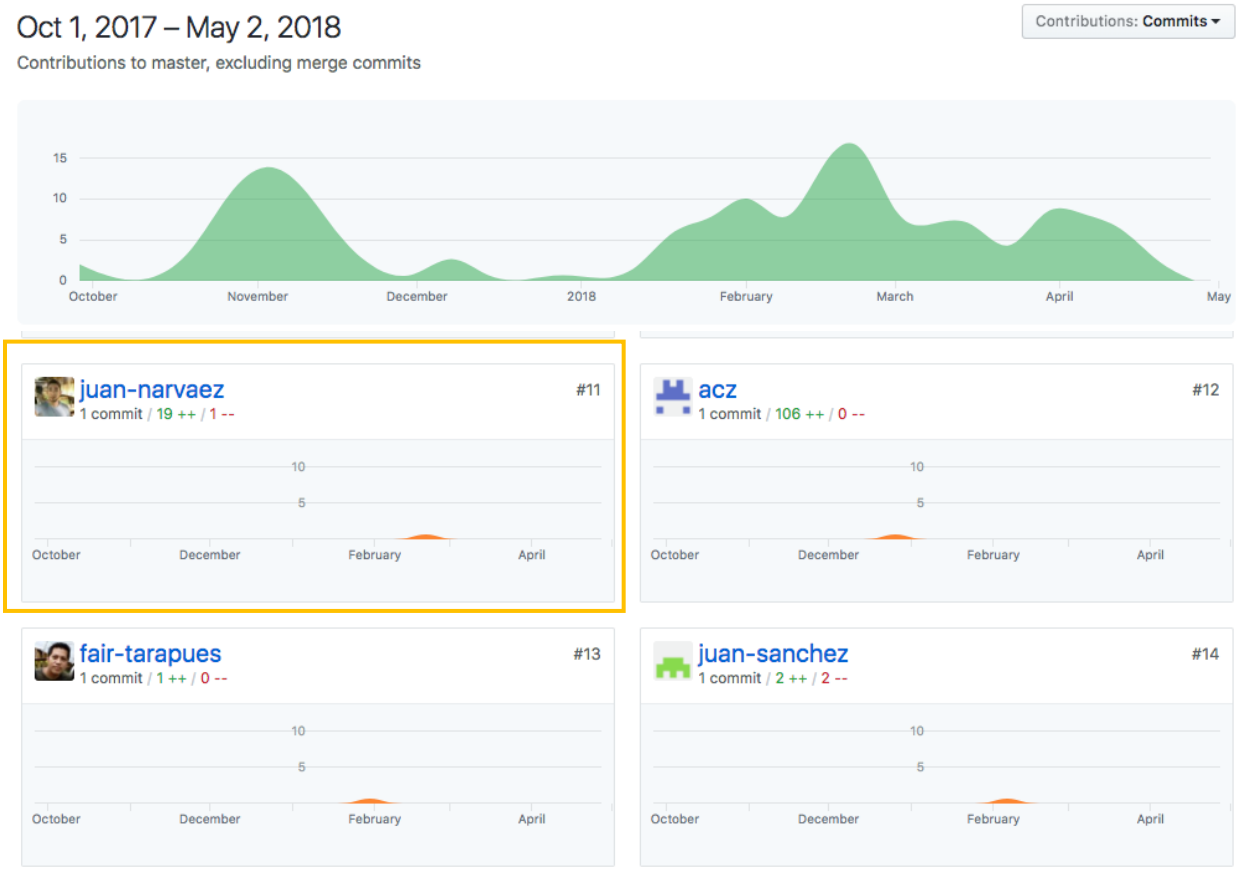
[28] Robert C. Martin, *Codigo Limpio: Manual de estilo para el desarrollo agil de Software*. España, 2014.

Anexo: Evidencias

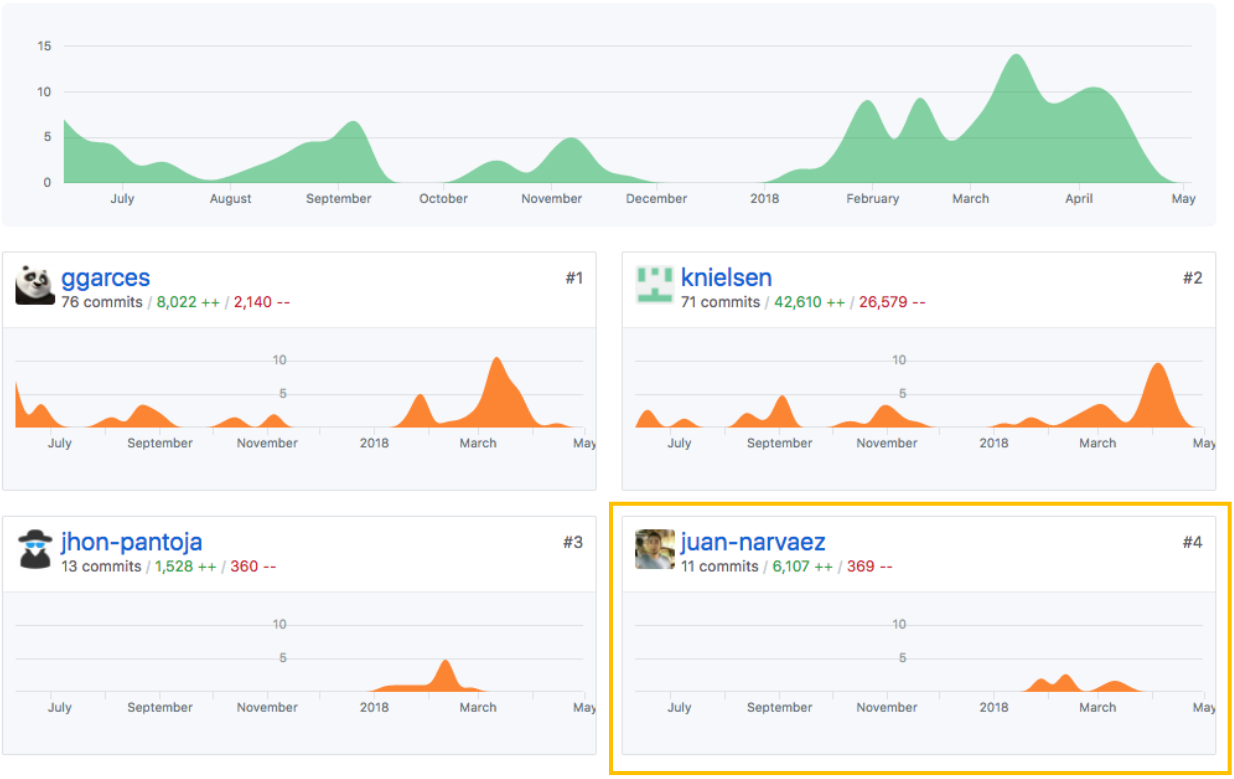
Contribuciones al repositorio *al-ui-threat-manager:*



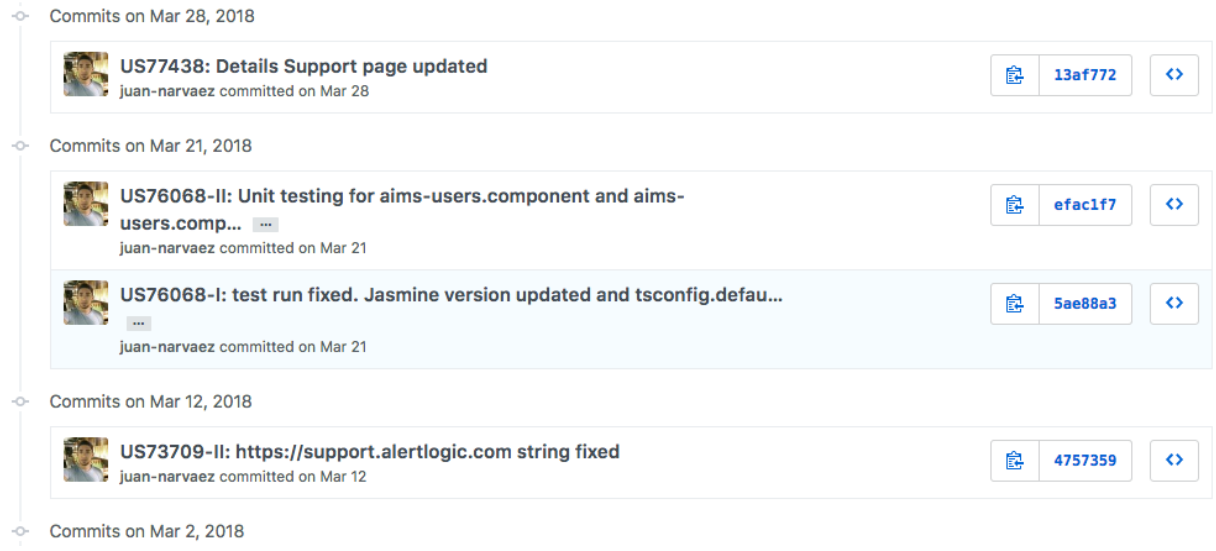
Contribuciones al repositorio *UI-Metadata:*



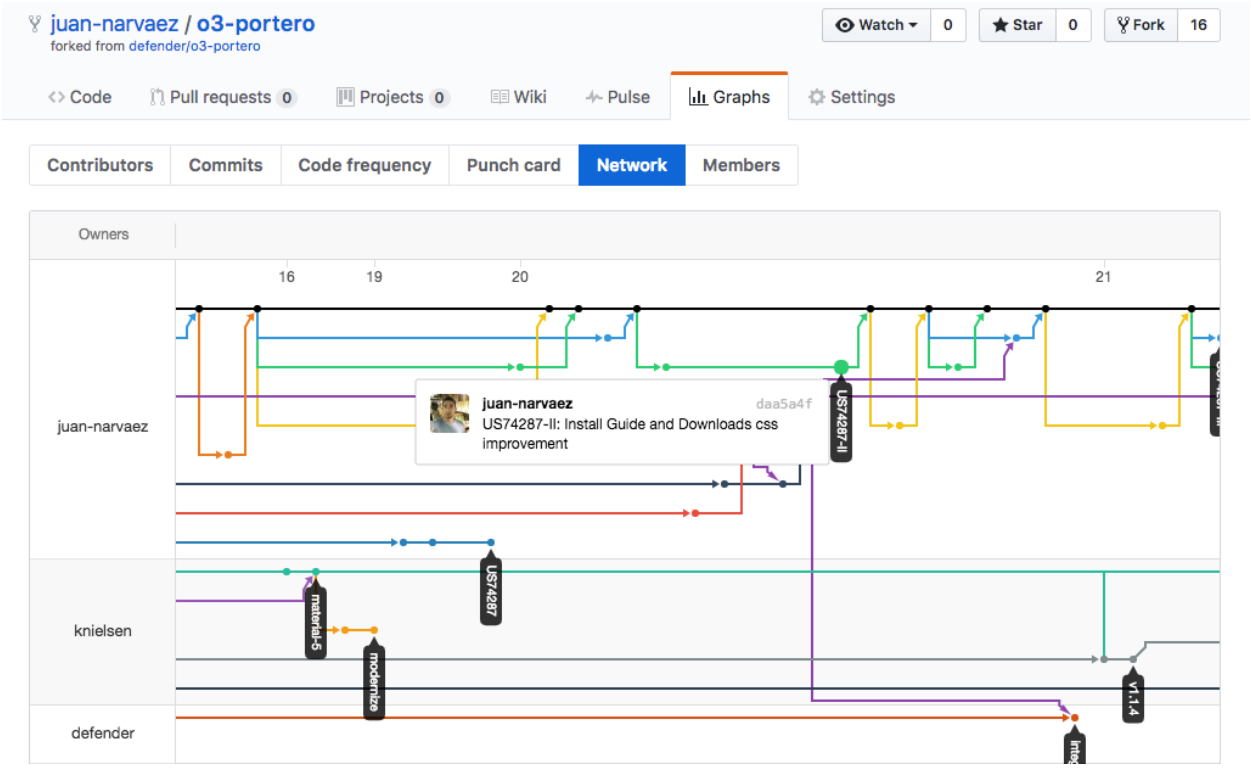
Contribuciones al repositorio *O3 Portero*:



Evidencia ultimos commits *O3 Portero*:



Ramas *O3 Portero*:




SECURITY
AWARENESS
Certificate of Achievement
Juan Narvaez
has successfully completed
KEVIN MITNICK SECURITY AWARENESS TRAINING 2016
KnowBe4
Security Awareness Training

ALERT LOGIC®

CERTIFICATE OF COMPLETION

PRESENTED TO

Juan Camilo Narvaez Sandoval

CC. 1.143.855.304

For successfully completing the 64 hours of

AngularJS Course

Santiago de Cali, Colombia

June 23, 2017

Issued by

ALERTLOGIC
Security. Compliance. Cloud
NIT 900.566.093-8

Diego Alejandro Ruiz Caicedo
Manager, Cloud Defender UI Services
Alert Logic Inc. Colombia