Universidad de la República

Facultad de Ingeniería, Universidad del Trabajo del Uruguay

Proyecto de grado para pustular a

Tecnólogo en Informática

Sistema de monitoreo de aplicaciones web e infraestructura

Autores

Camacho Possamai, Mauricio Gabriel

Deber Ospitales, Federico

Messina Fernandez, Jorge Gonzalo

Montes De Oca Gutierrez, Gonzalo Darvin

Sabatella Ferro, Martin Gerardo

Tutor

Ing. López, Montserrat

8 de julio de 2018

Resumen

El presente documento describe la investigación y el desarrollo de la creación de una herramienta de monitoreo de aplicaciones web e infraestructura de sistemas de información, propuesta en la asignatura 'Proyecto' como trabajo final de la carrera Tecnólogo en Informática.

Dada la naturaleza de los sistemas de información distribuidos, las aplicaciones se comunican a través de servicios sometidos a constantes cambios que llevan a resultados inesperados o fallidos. Por lo tanto, para prevenir estos errores de comunicación es necesario auditar el estado de los servicios y la infraestructura que lo soporta, administrar *logs*, configurar alertas y notificaciones. Por medio de este documento se expone el proceso de investigación que derivó en la búsqueda de estándares y tecnologías punteras disponibles para crear un recurso innovador que aporte al conocimiento.

Palabras clave

Logs, sitio, cliente, pryoxy reverso, Java EE, OWASP, MongoDB, MySQL, React Native, Android SDK, Angular, Websocket, Bootstrap, Docker, Vagrant, seguridad, encriptar, *backup*, API, XML, CSV, JSON, URL, TCP, UDP, REST, SNMP.

Historial de versiones

Fecha	Versión	Descripción	Autor
07/05/2018	1.0, master 32048eb	Elaboración del documento	Grupo 1
25/06/2018	2.0, master b7ac9c5a	Totalidad de requerimientos	Grupo 1
03/07/2018	2.1, master 24cb71ae	Correcciones documento	Grupo 1
08/07/2018	2.2, master 9a1bbd62	Documento finalizado	Grupo 1

Agradecimientos

Dedicamos el presente trabajo como agradecimiento a nuestras familias y amigos, por el apoyo en estos años de formación. Además a todos aquellos que a lo largo de la vida nos han motivado a cumplir nuestros objetivos. Gracias a los funcionarios que hacen posible la carrera y todos los que colaboraron indirectamente en este documento, dando su opinión o aportando con alguna idea o referencia. Un particular agradecimiento a Monserrat López, quien fue nuestra tutora y la responsable de gestionar esta última etapa de nuestra carrera.

Índice general

Capítulo 1

Introducción

En el contexto actual, donde las empresas utilizan múltiples tecnologías que se encuentran en continuo desarrollo, donde las fuentes de información son nodos en una red descentralizada de la cual dependemos de su disponibilidad para consultarlas, es necesario que las organizaciones tomen el control de sus recursos informáticos automatizando los procesos de monitorización. El descuido de un recurso informático puede causar pérdidas materiales e inmateriales en las organizaciones.

A continuación se plantea el proceso de desarrollo de una aplicación disponible en plataformas web y móvil, para el monitoreo de aplicaciones web y análisis de *logs*, administrable mediante un panel de control (*dashboard*) donde se podrán configurar alertas y notificaciones por diferentes plataformas de comunicación (correo electrónico, sms [32], *Whatsapp* [90], *Slack* [71], otros). Para lograr este objetivo fue necesaria la investigación de diferentes tecnologías y plataformas existentes a nivel internacional, de esta forma se logró una referencia clara para comenzar a plantear la solución.

Para probar la solución se configuró un proxy reverso [53] con diferentes aplicaciones web desplegadas, las cuales se registran en el sistema para ser monitoreadas.

También se cumple con requisitos de usabilidad y amigabilidad del sistema, dado que es necesario que el usuario pueda detectar rápidamente cualquier problema que se presente con sus aplicaciones web o recibir una notificación de su infraestructura de forma amena.

Capítulo 2

Objetivos planteados y resultado esperado

El objetivo del proyecto consiste en analizar y desarrollar una aplicación para la administración de páginas web e infraestructura informática.

Para ello es necesario realizar una investigación con el propósito de identificar el estado actual de tecnologías y documentación relacionadas al tema, y de esta forma intentar aportar nuevo conocimiento a partir de los avances hechos por la comunidad.

La aplicación estará disponible para plataformas web y móvil. Se deben tener en cuenta las mismas funcionalidades para ambas aplicaciones cliente *front-office*, además se requiere un módulo web *back-office* o perfil de administrador que permita la gestión completa del sistema.

Durante el proceso de desarrollo será necesario realizar pruebas sobre una infraestructura simulada mediante un *hardware* virtualizado, físico o *cloud* con sistema operativo Linux [80] donde se servirá un proxy reverso como punto de acceso a las distintas aplicaciones web a monitorear.

Se espera que se pueda configurar las aplicaciones web e infraestructura y recibir *logs* de aplicaciones, sistemas operativos y bases de datos, a los que se le aplicarán filtros y formato para su visualización gráfica en un panel principal o *dashboard*. Como solución se espera contar con la aplicación funcionando en base a la problemática planteada.

Capítulo 3

Estado del arte

En este capítulo se expresa formalmente la investigación realizada sobre las tecnologías aplicadas; el análisis de documentos, referencias, guías, *software* de utilidad y aplicaciones similares referentes al tema de estudio. Se presenta un marco conceptual que servirá de base teórica para la comprensión y desarrollo del proyecto.

Con el fin de acotar y orientar la investigación es necesario identificar en la propuesta factores que pueden ser considerados como hipótesis o las bases del posterior desarrollo. Es requerido que la aplicación sea desarrollada en Java Enterprise Edition [54], que cuente con un cliente móvil Android SDK [4] y un cliente web desarrollado en Angular [6] y Bootstrap [82]. Estas tecnologías si bien son fundamentales para el desarrollo, no serán el objeto de estudio principal, aunque se analizarán para comprender su estado actual y proyección a futuro comparado a herramientas similares.

Para recrear un ambiente de simulación de aplicaciones web e infraestructura, será de necesidad la instalación y configuración de un proxy reverso en donde estarán servidas las aplicaciones web e infraestructura. Por lo tanto se agrega a la hipótesis las tecnologías sugeridas para este propósito: Apache reverse proxy [7] y Nginx reverse proxy [49].

Antes de comenzar a investigar sobre el tema en cuestión se debe disponer de un alcance de lo que se pretende lograr para establecerlo como el mínimo aporte a realizar, de esta forma a consecuencia de la investigación poder ofrecer características de carácter innovador.

A continuación se repasan los puntos que determinan el alcance según lo expuesto en el apartado Resumen de la propuesta de proyecto: disponible como plataforma web, disponible como plataforma móvil, configurar y monitorizar aplicaciones web, configurar y monitorizar infraestructura, envío de *logs* en tiempo real, parsear *logs* de aplicaciones, sistema operativo y datos, aplicar filtros a los *logs* para su visualiza-

ción en el panel general o *dashboard*, configurar alertas (ver, limpiar, actualizar, etc), configurar alertas mediante notificaciones por correo electrónico y algún otro medio (sms, *slack*, *whatsapp*, entre otros).

Ahora con un alcance general de lo que se pretende, se proseguirá a investigar la existencia de tecnologías que cumplan con un objetivo similar, pero no sin antes listar los títulos de las sugeridas en la propuesta para integrarlas como parte de nuestra hipótesis.

A continuación se repasan las tecnologías propuestas para comenzar la investigación: Zabbix [95], New Relic [48], Rsyslog [70], Nagios [47], Elastic Stack, Elasticsearch [16], Kibana [35], Logstash [20] y Filebeat [18].

Expuesta la hipótesis, solo resta comenzar la investigación formal con el fin de desarrollar la idea e introducirnos en el tema.

3.1. Marco teórico

En la actualidad las organizaciones se apoyan cada vez más en sistemas de información, eventualmente ante un fallo a cualquier nivel, es probable que genere pérdidas materiales e inmateriales que amenacen la estabilidad de la empresa. Unido a esto, con el paso del tiempo la complejidad de las infraestructuras es cada vez mayor, muchas veces las organizaciones se preparan para responder a estas excepciones mediante **solución reactiva** (recurrir a *backups*, réplicas, etc.).

Al monitorizar la infraestructura y aplicaciones es posible realizar **acciones preventivas** que eviten llegar al punto de aplicar una solución reactiva, de esta forma decrecen las posibilidades de fallo, lo que garantiza la continuidad y disponibilidad de los servicios.

Es posible monitorear casi cualquier elemento de infraestructura (*hardware*, sistemas operativos, espacio en disco, disponibilidad de aplicaciones, ancho de banda, temperatura, etc). Por lo tanto cuanto más procesos sean auditados menor será la probabilidad de aplicar soluciones reactivas, lo que en la organización se traduce en mantener sus activos críticos controlados evitando pérdidas. La solución propuesta será abordada para que cumpla una función preventiva ante amenazas expuestas en este marco teórico.

3.2. Investigación realizada

Se comienza la investigación en concordancia con lo expuesto en anteriormente en este capítulo, por ello primeramente se revisó el estado actual de tecnologías vinculadas al objeto de estudio.

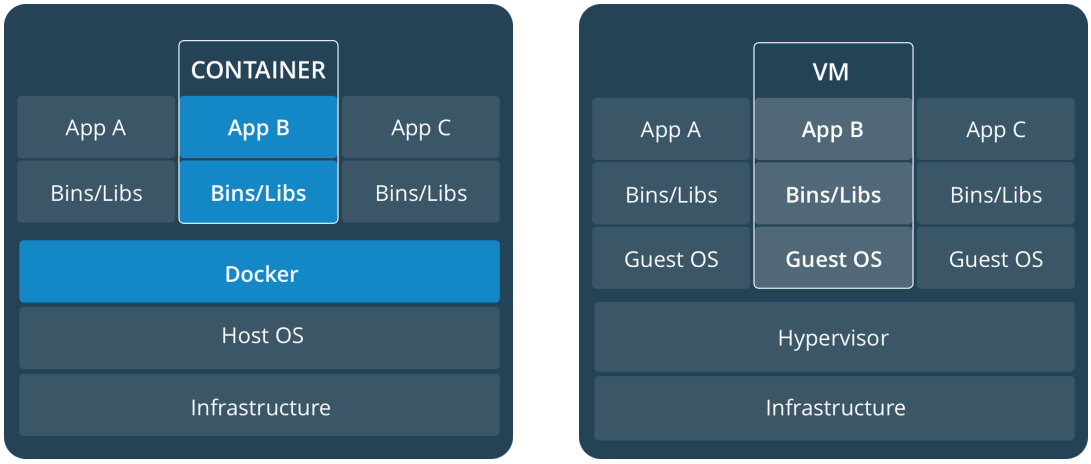
3.2.1. Normalizando el entorno de desarrollo

La experiencia indica que normalizar el ambiente puede ser una tarea compleja de coordinar. Es de conocimiento que el recurso de *hardware* en las computadoras personales en ocasiones puede ser insuficiente como para soportar el sistema operativo base, más una maquina virtual con linux [80] y el resto de *software* de desarrollo. Para optimizar los recursos se normalizó el sistema operativo a Ubuntu 16.04 LTS [83] y se instaló directamente sobre el *hardware* físico.

Sobre la virtualización

Conociendo la existencia de herramientas como Docker y Vagrant [27], aunque sin experiencia en su uso y con un concepto ambiguo, se prosiguió a analizar las características de cada herramienta, llegando a la conclusión de que cumplen propósitos distintos. Con respecto a Vagrant, es una herramienta orientada a armar entornos de desarrollo sobre un proveedor de máquinas virtuales como Virtualbox [60], Vmware [85] o Hyper-v [84]. En cambio Docker es una herramienta para crear contenedores de *software* ligeros y portables con el fin de facilitar el despliegues de aplicaciones [27]. Como cuentan con diferencias en las arquitecturas se pueden utilizar de forma complementaria.

En la Figura 3.1 se muestra una comparación entre contenedores y máquinas virtuales.



(a) Contenedores (b) Maquinas virtuales

Figura 3.1: Comparación de contenedores y maquinas virtuales

Sobre la tecnología backend

Actualmente se encuentra disponible la versión 12.0.0.Final del servidor de aplicaciones Wildfly [68], en relación a este hecho JavaJDK se encuentra en la versión 9.0.4 [61]. Se realizaron pruebas usando Wildfly 10.1.0.Final con JavaJDK 9.0.4 obteniendo una advertencia de incompatibilidad al ejecutar el servidor de aplicaciones la cual no impidió su funcionamiento, aunque este tipo de combinaciones seguramente no sea conveniente a futuro. Salvo este tipo de excepciones por experiencia de los autores estas tecnologías son muy estables por lo que sera recomendable instalar las ultimas versiones disponibles, tal como recomiendan los creadores del *software*.

Sobre la tecnología *frontend*

En cuanto a tecnología *frontend*, será de vital importancia el uso de *frameworks* javascript [89] para webs de una sola carga o SPA(*Single Page Aplication*), Angular [6] y React [65] actualmente tienen mucha popularidad en la comunidad, además cuentan la ventaja de extenderse al desarrollo de aplicaciones móviles nativas con el uso de Ionic [34] y React Native [66] respectivamente.

Opcionalmente para complementar existen *frameworks* de estilos HTML [88], CSS [87], Javascript [89] como Bootstrap v4.0.0 [82] o Materialize 1.0.0-beta [38].

Para emular la aplicación móvil se puede usar Android Studio [3] u opcionalmente Android Debug Bridge (ADB) [2] es una herramienta de líneas de comandos versátil que

te permite comunicarte con una instancia de un emulador o un dispositivo Android conectado.

3.2.2. Simulación de aplicaciones web e infraestructura

Para simular un entorno de aplicaciones web e infraestructura será de utilidad la configuración de un servidor con proxy reverso y servicio web donde se desplegarán las aplicaciones a monitorear.

Sobre proxy reverso

Un proxy reverso [53] aparece para el cliente como un servidor web ordinario. No es necesaria ninguna configuración especial en el cliente, simplemente realiza solicitudes ordinarias de contenido en el espacio de nombres del proxy reverso y este decide dónde enviar esas solicitudes devolviendo el contenido como si fuera el origen.

En la Figura 3.2 se muestra el diagrama de proxy reverso.

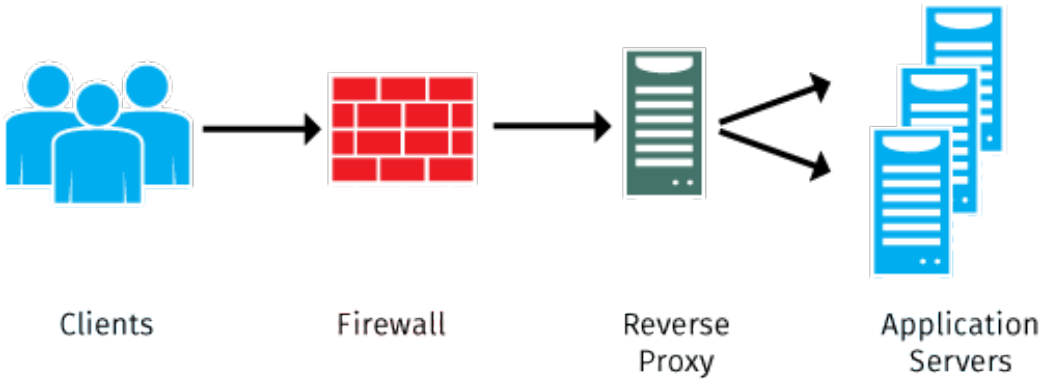


Figura 3.2: Diagrama de proxy reverso

Un uso típico de un proxy reverso es proporcionar acceso a los usuarios de Internet a un servidor que está detrás de un *firewall* [14]. También se pueden usar para equilibrar la carga entre varios servidores de servicios de fondo o para proporcionar el almacenamiento en caché de un servidor de fondo más lento. Además, se pueden usar simplemente para traer varios servidores al mismo espacio de URL [91].

Sobre Apache httpd

El servidor Apache HTTP, es un servidor web de propósito general, poderoso y flexible. Implementa los últimos protocolos incluyendo HTTP/1.1 (RFC2616) [76]. Es altamente configurable y extensible a través de módulos de terceros desarrollados usando el *Apache module API* [74]. Además se proporciona el código fuente completo y viene con una licencia no restringida de la *Apache Software Fundation* [75]. Corre en Windows [43] y la mayoría de las versiones de Unix [78].

Algunas de las ventajas que tiene esta herramienta son: es altamente configurable, modular, completamente gratuito, multi-plataforma (compatible con Windows, Linux, y MacOS), extensible a través de módulos, popular (comunidad, ayuda/soporte), alto rendimiento, seguridad SSL [33] y TLS [31].

Por otro lado, tiene las siguientes desventajas: falta de integración, posee formatos de configuración no estándar y no cuenta con interfaz gráfica para su gestión.

El Servidor Apache HTTP puede configurarse tanto en un modo de proxy directo como reverso (también conocido como puerta de enlace). Apache puede extender la característica de proxy reverso a través de la extensión del módulo proxy.

Sobre Nginx

Nginx es un servidor web HTTP/proxy reverso, ligero y de alto rendimiento y un servidor proxy de correo electrónico, originalmente escrito por Igor Sysoev. Es *software* libre y de código abierto licenciado bajo BSD simplificada [37]. También existe una versión comercial bajo el nombre de Nginx Plus. Es multiplataforma, corre en sistemas de tipo Unix y Windows.

Las ventajas que tiene Nginx son: es estable, sencillo, performante, seguro, gratuito, alto rendimiento, escalabilidad y tiene soporte comercial.

Las desventajas que tiene son: no es integrado, posee adaptador propio y las herramientas gráficas de gestión pueden ser costosas.

A diferencia de Apache, Nginx es un proxy reverso por definición por lo que no requiere de la integración de un módulo para adquirir esta característica.

3.2.3. Aplicaciones similares

Se realizó una investigación sobre las aplicaciones de monitoreo de aplicaciones web e infraestructura ya existentes en el mercado. Cabe resaltar que no se encontraron aplicaciones similares a nivel nacional. Es necesario aportar un enfoque nuevo porque se compite con productos de carácter internacional.

Durante dicha investigación se detectaron algunas características similares entre ellas, como por ejemplo la utilización de gráficas y reportes bastante intuitivos para el usuario.

A continuación se detallan las diferentes funcionalidades así como también las ventajas y desventajas que tienen las principales aplicaciones de monitoreo que existen en el mercado hoy en día.

Filebeat

Es una herramienta para el envío de los logs. Si bien Logstash ya realiza esta tarea, tiene el problema que al hacerlo consume demasiados recursos, haciendo el ELK lento, siendo la JVM uno de los causantes de este problema. Para resolverlo se desarrollaron herramientas como Filebeat que se integra a la perfección con ELK mejorando significativamente la performance.

En la Figura 3.3 se muestra la plataforma Beat y el despacho de logs.

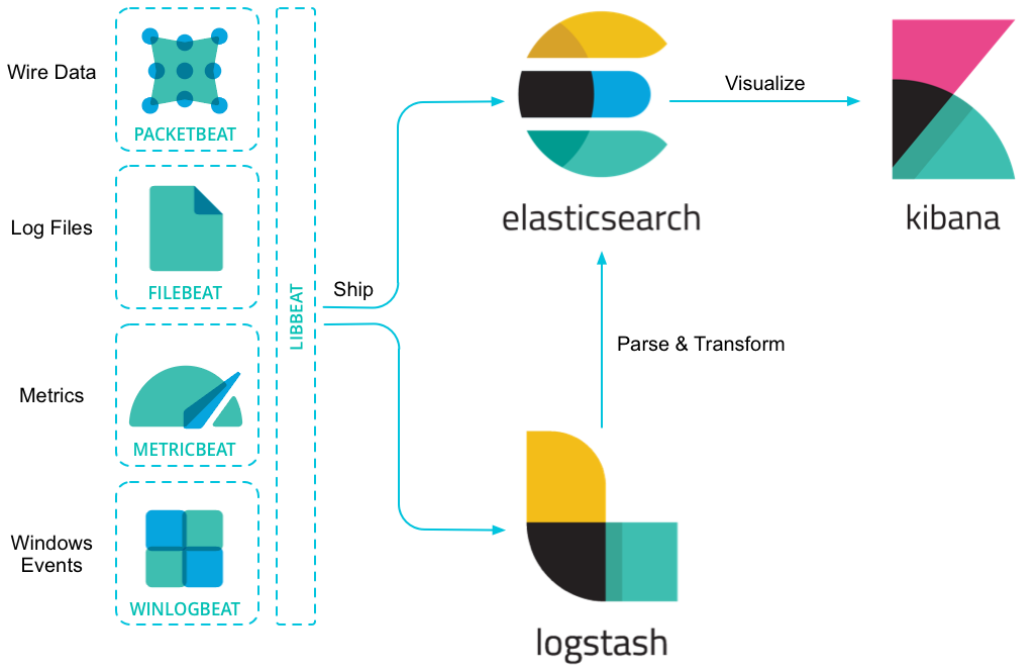


Figura 3.3: Plataforma Beat y despacho de logs

Logstash

Es una herramienta desarrollada por Elastic y que funciona bajo la JVM de Java [56]. Ésta nos permite administrar los *logs* de nuestras aplicaciones, de manera que podemos usarla para recolectar, parsear y guardar los logs para búsquedas posteriores.

Esta herramienta basa su funcionamiento en la integración de entradas, *codecs*, filtros y salidas. Las entradas son las fuentes de datos que se usarán posteriormente; los *codecs* convierten un formato de entrada en otro que Logstash acepte, y éste último formato en otro de salida. Los *codecs* se usan (normalmente) cuando los datos no son texto plano.

Los filtros son acciones usadas para procesar eventos, modificarlos o eliminarlos. Por último, las salidas son los destinos donde se enviarán los datos procesados.

La gran diferencia de Logstash con el resto de herramientas del estilo reside en que tiene a su disposición un gran número de *plugins* para las tres partes de las que se compone (ELK) [17].

En la Figura 3.4 se muestra la arquitectura de Logstash.

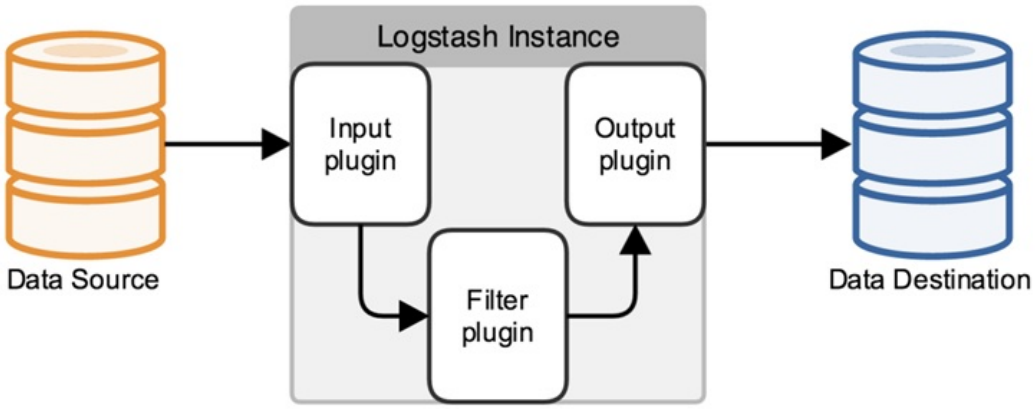


Figura 3.4: Arquitectura Logstash

Cabe mencionar que Logstash puede configurarse para utilizar varios servidores de la siguiente forma: se enviarán los *logs* a un único servidor hasta que éste falle. Cuando eso ocurra, cambiará de servidor y listo. Sin embargo, los *logs* almacenados en el servidor que ha fallado no serán accesibles hasta la recuperación del mismo.

En la Figura 3.5 se muestra el esquema de Elastic Stack.

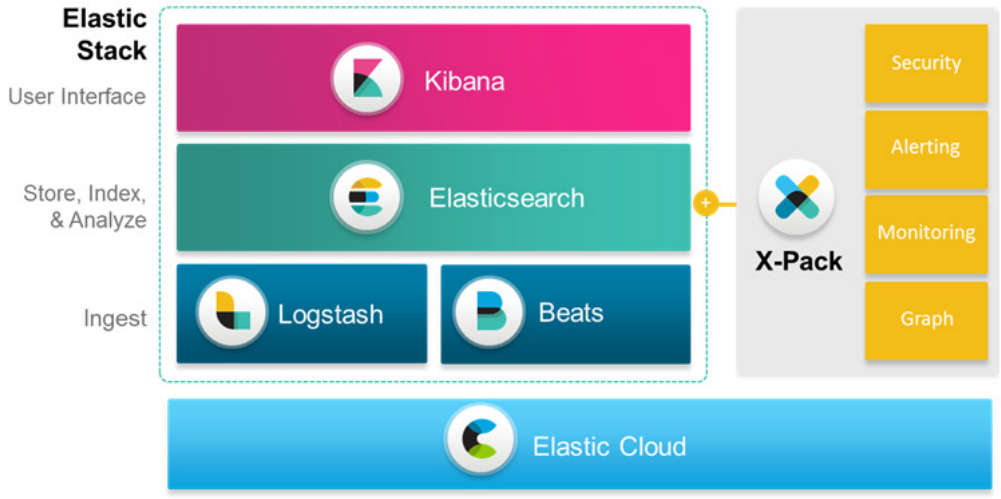


Figura 3.5: Esquema Elastic Stack

Elasticsearch

Es un servidor de búsqueda basado en Apache Lucene [73]. Provee un motor de búsqueda de texto completo, distribuido y con capacidad de multi-tenencia con una interfaz web RESTful [94] y con documentos JSON [86].

Elasticsearch está desarrollado en Java y está publicado como código abierto bajo las condiciones de la licencia Apache [72].

Las ventajas que provee Elasticsearch son: al estar desarrollado en Java es compatible en todas las plataformas donde Java lo sea, tiene una gran velocidad de respuesta, es distribuido y ésto lo hace fácilmente escalable y adaptable a las distintas situaciones, realiza respaldos de los datos almacenados, es fácil de invocar desde varios lenguajes de programación dado que utiliza objetos JSON como respuesta.

Las principales desventajas son: sólo soporta como tipos de respuesta JSON, lo que lo limita al no soportar otros lenguajes, como CSV [29] o XML [92]. Algunas situaciones pueden generar casos de *"split brain"*.

En la figura 3.6 se muestra un esquema con los componentes de Elastic.

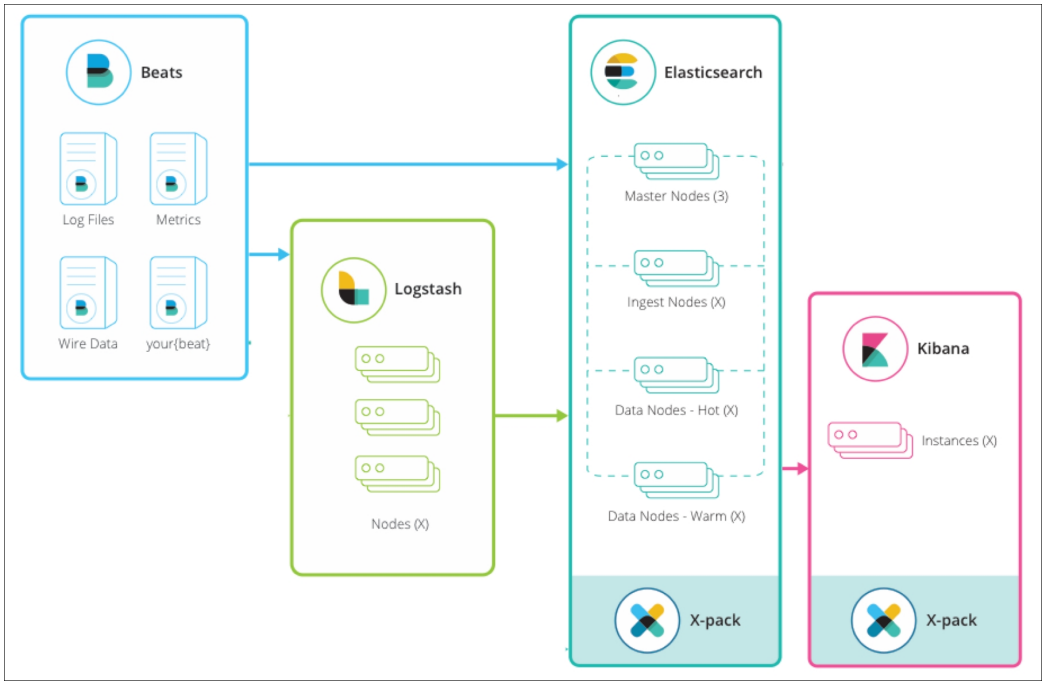


Figura 3.6: Componentes Elastic

Kibana

Kibana es un sistema analítico de código abierto perteneciente a Elastic, que nos permite visualizar datos y realizar búsquedas sobre Logstash y otros conjuntos de datos indexados en ElasticSearch. Kibana es un *plugin* de Elasticsearch, y conforma el *Stack* ELK.

Este *plugin* ofrece una interfaz muy potente sobre Logstash pero también permite crear *dashboards* a medida, con características como personalización, selección de rangos y además poder compartir y guardar.

Ofrece la posibilidad de crear componentes gráficos, hasta crear *dashboards* completamente personalizados.

No cuenta con un cliente móvil.

En la Figura 3.7 un ejemplo de la interfáz gráfica de Kibana.

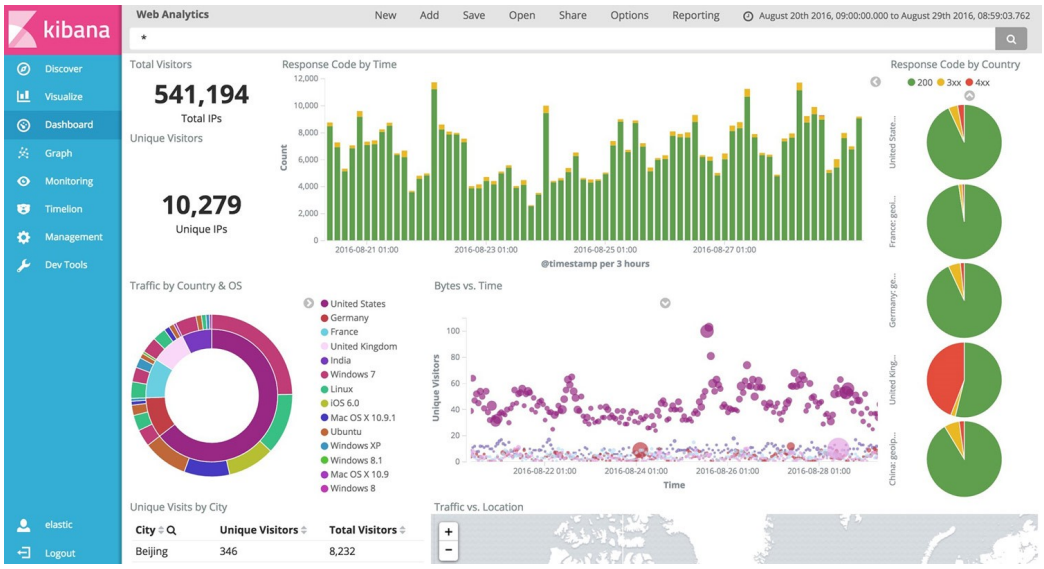


Figura 3.7: Interfáz gráfica Kibana

Zabbix

Zabbix es un sistema para monitorear la capacidad, el rendimiento y la disponibilidad de los servidores, equipos, aplicaciones y bases de datos. Además ofrece características avanzadas de monitoreo, alertas y visualización.

La aplicación se instala en un servidor dedicado a recolectar información. Luego se instalan los agentes Zabbix en cada equipo o estación de trabajo que se desee monitorear y quedan a la espera de las órdenes del servidor recolector Zabbix. Los agentes envían únicamente la información que les pida el servidor.

Los agentes pueden ser instalados en la mayoría de sistemas operativos existentes en el mercado (Linux, Mac, Windows, AIX [28], Solaris [59]) y son muy livianos y consumen un mínimo de recurso del equipo donde se instale.

Por otro lado, Zabbix también ofrece la posibilidad de monitorear sin necesidad de agentes. Esto lo realiza a través de protocolos SNMP [13] (*Simple Network Management Protocol*) y TCP [26] (*Transmission Control Protocol*).

Algunas de las características que ofrece Zabbix son: monitoreo centralizado a través del administrador web, disponible para diferentes sistemas operativos (Linux, Windows, Mac OS, entre otros), envío de alertas vía correo electrónico, alto rendimiento y capacidad de monitoreo de dispositivos (servidores, *hardware* como impresoras, *routers*, entre otros), monitoreo centralizado a través del administrador Web (*routers*front-end) y agentes que pueden instalarse en sistemas basados en Unix y Windows.

Ventajas de utilizar Zabbix: interfaz basada en la web, reportes detallados, configuración sencilla, estadísticas en tiempo real del estado de los servidores, reduce los costos de operación al evitar el tiempo de inactividad y tiene una baja curva de aprendizaje.

La desventajas que posee Zabbix son: requiere de la instalación de numerosos *plugins* para ser eficiente y poder alcanzar sus funcionalidades completas, no cuenta con una librería oficial de *plugins* para la comunidad, no tiene la posibilidad de trabajar con herramientas Enterprise como Oracle [58], Active Directory [40] o Exchange [41].

En la Figura 3.8 se muestra cómo es el *Dashboard* de Zabbix.

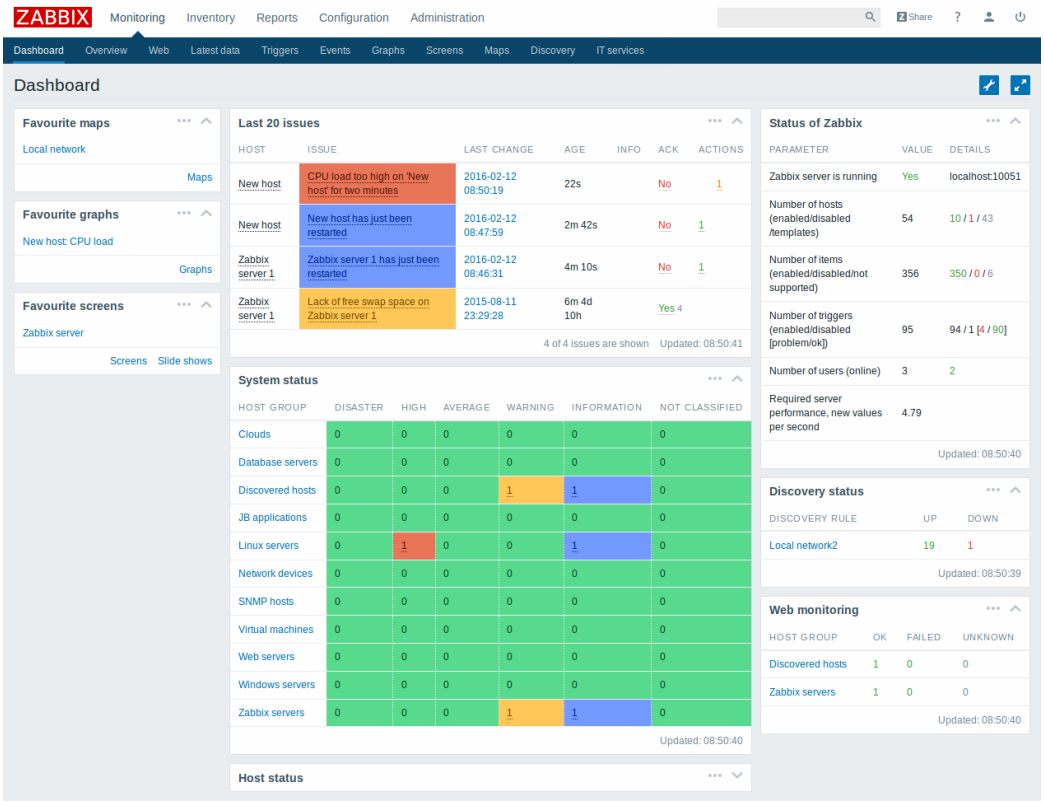


Figura 3.8: *Dashboard*, Zabbix [95]

En la Figura 3.9 se muestra un ejemplo de cómo se visualizan las gráficas de estado de infraestructura en Zabbix.

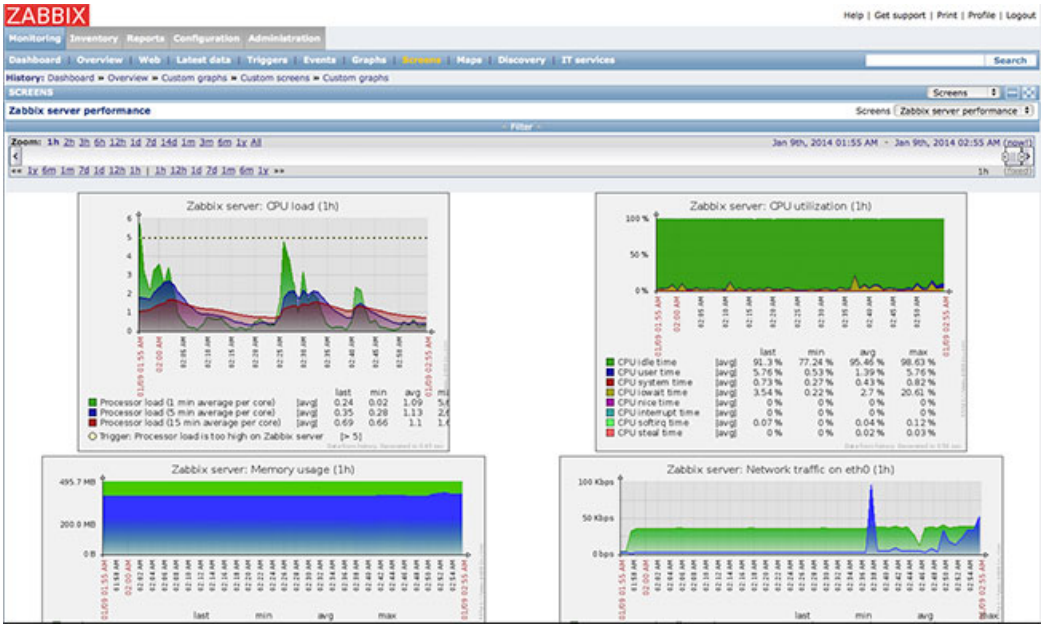


Figura 3.9: Gráficas de estado de infraestructura de Zabbix

Oficialmente Zabbix no cuenta con un cliente móvil pero proporciona una API que es utilizada por muchos desarrolladores para crear aplicaciones móviles donde se puede visualizar el estado actual de la infraestructura o servicios que se quieran monitorear.

Rsyslog

Es un sistema rápido y eficiente para el procesamiento de *logs*, ofrece alto rendimiento, excelentes características de seguridad y un diseño modular. Permite el ingreso de datos desde diversas fuentes, transformación de datos y salida de resultados hacia varios destinos.

Para su funcionamiento se configura un servidor central que recolecta la información de los *logs* que son enviados desde los equipos conectados a la red.

Sus características principales son: filtra cualquier parte del mensaje Syslog [63], es multi-hilo, formato de salida totalmente configurable, compatible con los protocolos TCP, SSL, TLS, RELP [64], permite almacenar los *logs* en archivos de texto simple o bases de datos MySQL [57], Oracle y PostgreSQL [79], en sistemas Linux viene instalado por defecto.

En la Figura 3.10 se muestra un ejemplo del esquema de comunicación de Rsyslog.

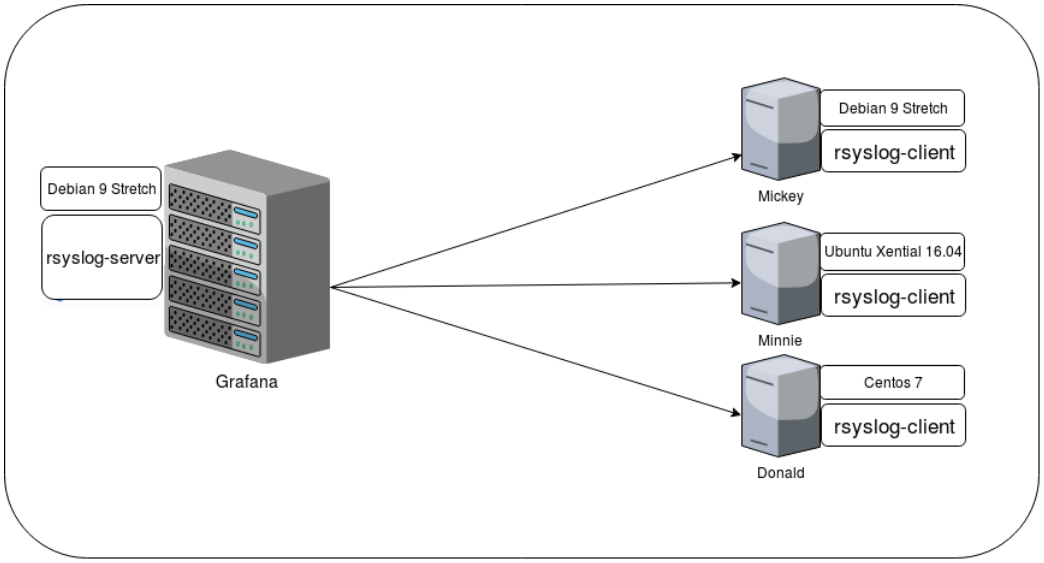


Figura 3.10: Esquema de comunicación de Rsyslog

Reporta mensajes de diferentes tipos, a su vez cada uno de éstos tiene asociado un nivel de prioridad, dependiendo de la gravedad del mensaje en el *log*.

A continuación se detallan los tipos de mensajes de Rsyslog ordenados de mayor a menor de acuerdo a su nivel de prioridad. EMERG (sistema inutilizable), ALERT (requiere intervención inmediata), CRIT (condición crítica), ERR (condición de error), WARN (advertencia, error potencial), NOTICE (las condiciones son normales pero el mensaje es importante), INFO (mensaje informativo) y DEBUG (mensaje de depuración).

En la Figura 3.11 se muestra un ejemplo del esquema de comunicación de mensajes de Syslog.

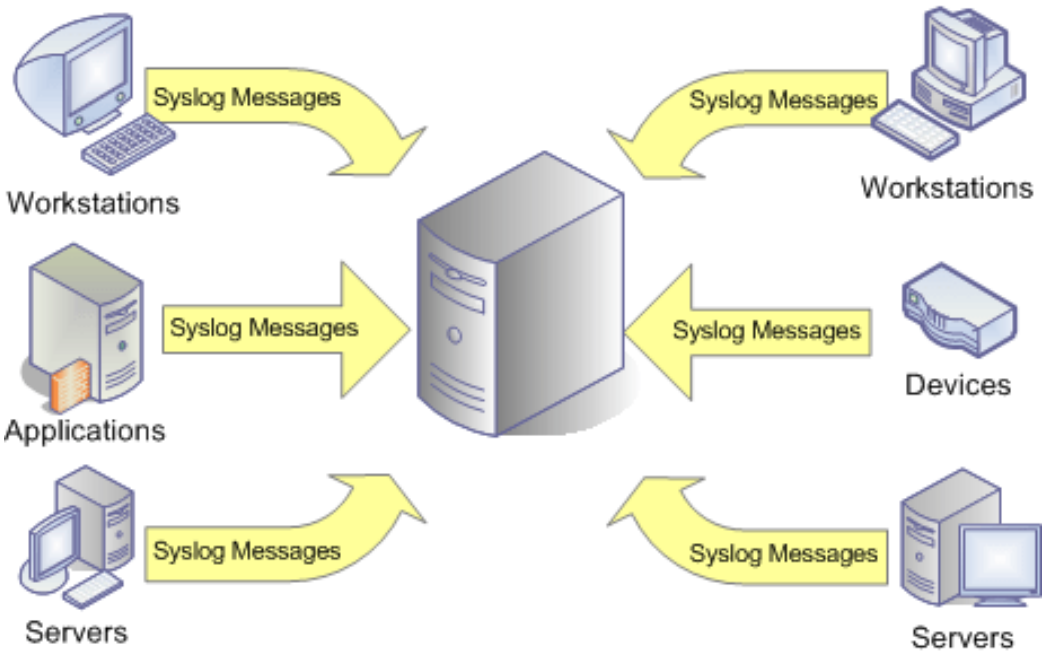


Figura 3.11: Esquema de comunicación de mensajes Syslog

New Relic

New Relic es un completo sistema de motorización de máquinas que permite tener un control en tiempo real de los recursos disponibles. Controla tanto aplicaciones web como aplicaciones móviles Android [5] e iOS [8]. Tiene soporte en la nube y se puede integrar con Docker. Ofrece tres versiones: LITE que es gratuita, estándar y profesional.

Permite simular usuarios (tanto flujo como interacciones) para anticiparse a los errores y usa el servicio de alertas para avisar de estos. También da una vista del servidor desde la perspectiva de la propia aplicación.

En la Figura 3.12 se muestra cómo se visualizan los gráficos de infraestructura de New Relic.

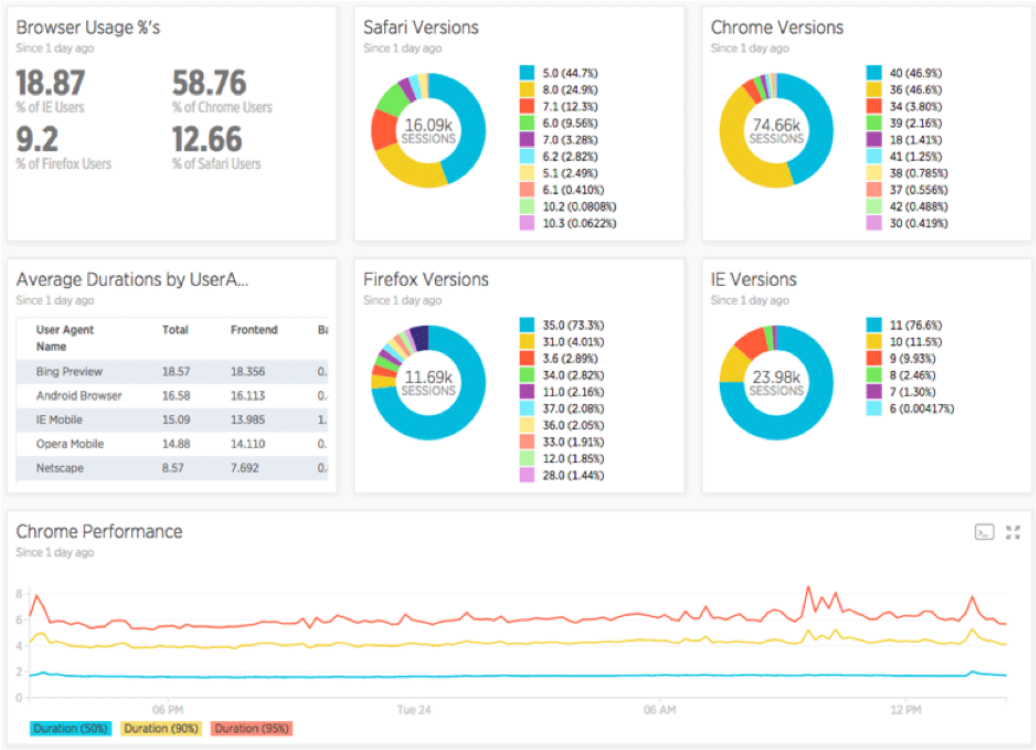


Figura 3.12: Gráficos de infraestructura de New Relic

Sus principales características son: monitorizar conexiones HTTP (tiempos de respuesta, número de peticiones), monitorización de errores (avisos cuando se detectan fallos de ejecución o conexión), fijar alertas sobre datos de referencia (tiempos de respuesta, errores de autenticación), estadísticas de rendimiento en distintos dispositivos (uso de memoria, velocidad de respuesta), estadísticas de usuarios que la usen según el sistema operativo utilizado, es multiplataforma.

Nagios

Nagios [47] es un sistema de monitoreo que permite a las organizaciones identificar y resolver problemas de infraestructura antes de que afecten procesos comerciales críticos.

En la Figura 3.13 se muestra la interfáz de Nagios.

Algunas de las características que provee Nagios son: monitoreo integral, visibilidad y conciencia, remediación de problemas, planificación proactiva, informes, capacidades *multi-tenant*, arquitectura extensible y código personalizable. A continuación se describen las características anteriormente nombradas.

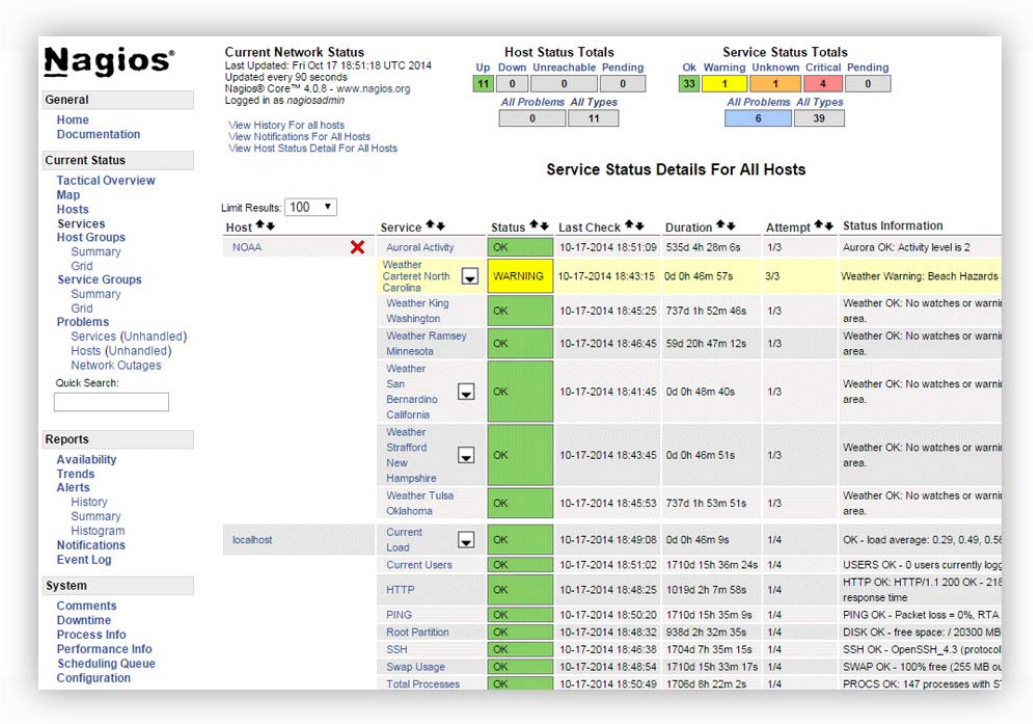


Figura 3.13: Interfáz de Nagios

Monitoreo integral: capacidad de monitorear aplicaciones, servicios, sistemas operativos, protocolos de red, métricas del sistema y componentes de infraestructura.

Visibilidad y conciencia: vista centralizada de toda la infraestructura, información detallada de estado disponible a través de la interfaz web, detección rápida de fallas, alertas por correo electrónico y/o SMS.

Remediación de problemas: los reconocimientos de alerta proporcionan comunicación sobre problemas conocidos. Los controladores de eventos permiten el reinicio automático de aplicaciones y servicios fallidos.

Planificación proactiva: los complementos de planificación de tendencias y capacidad garantizan que esté al tanto de la infraestructura obsoleta. El tiempo de inactividad programado permite la supresión de alertas durante las actualizaciones de infraestructura.

Informes: los informes históricos proporcionan un registro de alertas, notificaciones, interrupciones y respuesta de alertas. Los complementos de terceros amplían las capacidades de generación de informes.

Capacidades *multi-tenant*: el acceso *multi-tenant* a la interfaz web permite a los interesados ver el estado de la infraestructura. Las vistas específicas del usuario aseguran que los clientes vean solo sus componentes de infraestructura.

Arquitectura extensible: la integración con aplicaciones internas y de terceros es fácil con múltiples APIs [1]. Cientos de complementos desarrollados por la comunidad extienden la funcionalidad básica de Nagios.

Comunidad activa: las listas de correo de la comunidad activa brindan soporte gratuito. Cientos de complementos desarrollados por la comunidad amplían su funcionalidad principal.

Código personalizable: software de código abierto. Acceso completo al código fuente. Publicado bajo la licencia GPL [22].

En la Figura 3.14 se muestran algunas pantallas de la aplicación móvil de Nagios.

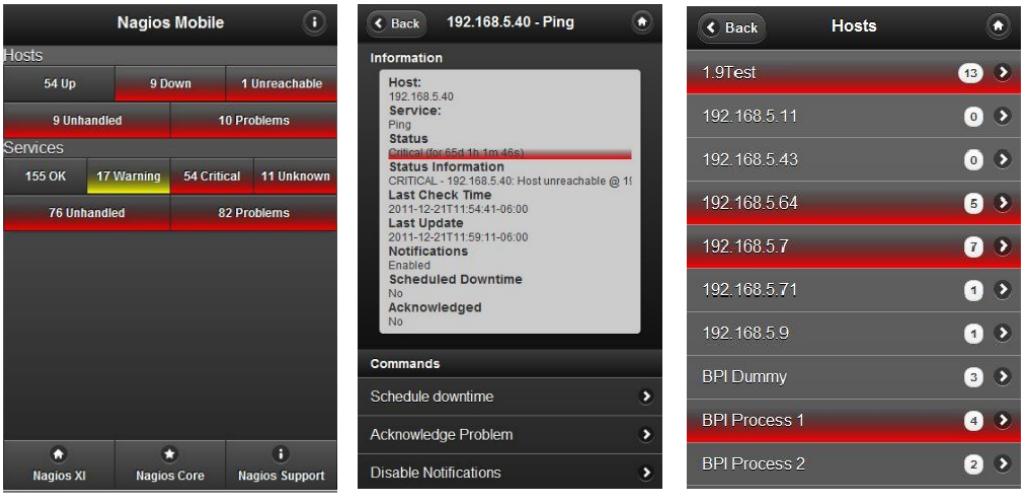


Figura 3.14: Pantallas de aplicación móvil de Nagios

Capítulo 4

Análisis del problema

Se debe implementar un sitio web y una aplicación móvil, que consiste de una herramienta de monitoreo de aplicaciones web e infraestructura de sistemas de información. Dicha herramienta deberá permitir a los usuarios tener un control de sus aplicaciones, mediante notificaciones que recibirá a través de alertas personalizadas. La interfaz debe ser amigable y usable, de forma que el cliente pueda visualizar fácilmente el estado de las aplicaciones a monitorear, ya sea el estado actual o en el período que desee. La representación de la información se realizará a través de gráficas que se actualizarán en tiempo real.

A la hora de realizar el análisis, lo que primero se identificó fueron los distintos requerimientos. Luego, en base a estos requerimientos, se llegó a un modelo de dominio que represente la realidad del sistema. También se identificaron los diferentes casos de uso, desarrollando los más críticos, entendiéndose por críticos los que representan un mayor reto para el equipo de desarrollo, debido a la falta de conocimiento de tecnologías o de procedimientos de implementación. Del análisis también se desprendió un modelo de diseño y otro de datos que se detallarán en los puntos a continuación.

4.1. Requerimientos del sistema

La requisitos del sistema sirven como medio de comunicación entre los involucrados del proyecto, en el se especifican las necesidades de la aplicación a desarrollar. Para ahondar en detalle ver en documento anexos: apéndice - Especificación de requerimientos de software.

4.1.1. Aplicación a desarrollar

Será una aplicación, que permitirá a los usuarios registrados en el sitio realizar el monitoreo de sus aplicaciones, suscribirse a las alertas para recibir las notificaciones, que pueden ser por ejemplo vía SMS o Email. También se desarrollará una aplicación para dispositivos con sistema operativo Android [5], que brindará la posibilidad de realizar las mismas operaciones que en la web. De esta forma se logrará que el cliente pueda acceder al sistema desde varios dispositivos y sin la necesidad de poseer una computadora.

Se crearán dos perfiles de usuario, uno de administrador y otro operador. El operador tendrá acceso a la visualización de las gráficas y los reportes, ver, comentar y resolver alertas. El administrador, además de las funciones del operador, podrá dar de alta nuevos usuarios, alertas y eventos.

4.2. Casos de uso

En esta sección se exhibirán los diagramas de casos de uso y también se comentarán los casos de uso críticos identificados durante el análisis de los requerimientos solicitados. Para ahondar en detalle ver en documento anexos: apéndice - Casos de uso.

4.2.1. Casos de uso identificados

En la Figura 4.1 se muestra en un diagrama general el alcance de los casos de uso identificados para resolver el sistema.

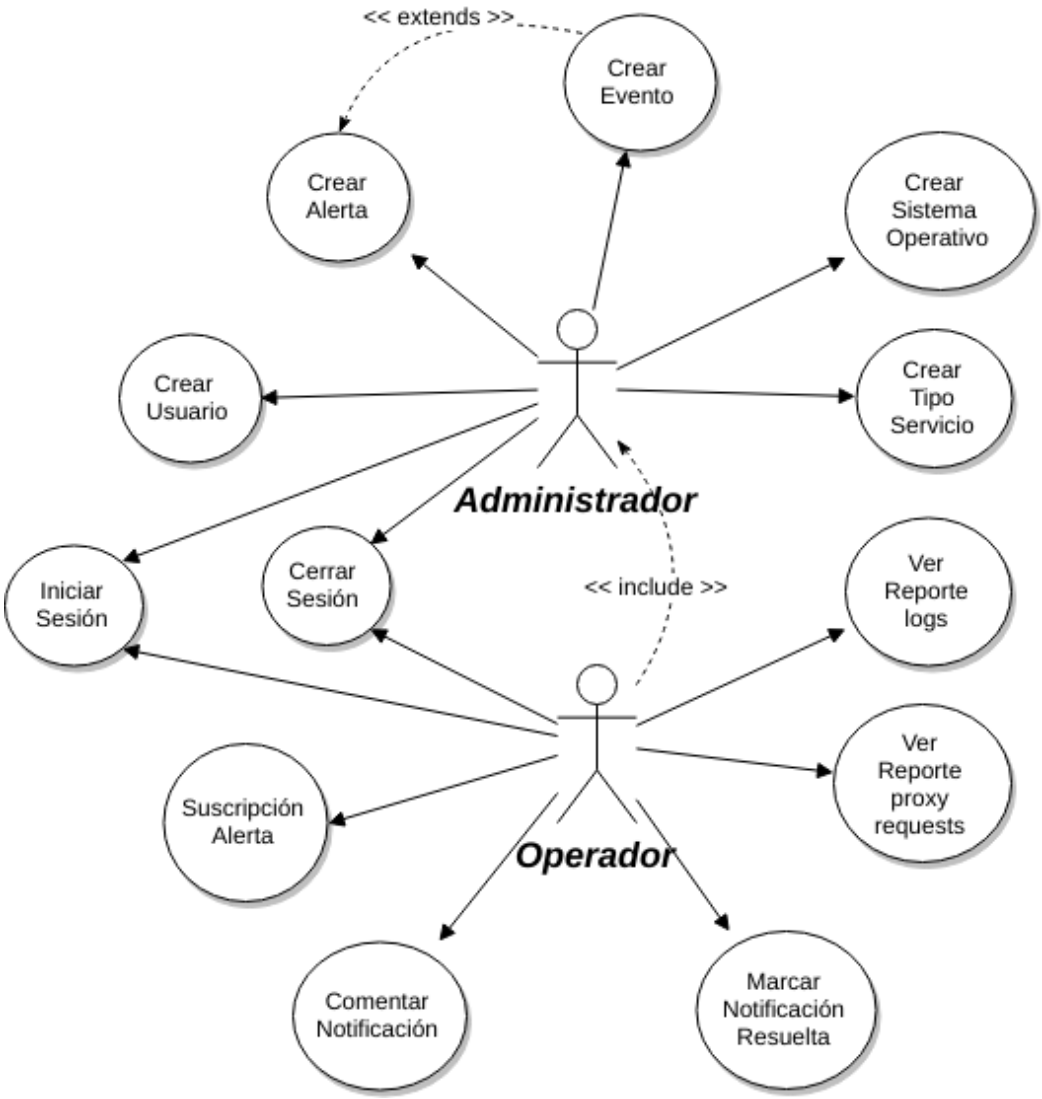


Figura 4.1: Diagrama de casos de uso identificados

4.2.2. Casos de uso críticos

En este apartado se explica en qué consisten los casos de uso identificados como críticos.

Crear evento

Este caso de uso es crítico porque sobre los eventos trabaja todo el sistema. Los eventos pueden ser de tipo servidor o servicio. Con ellos se registra la actividad de las aplicaciones, o infraestructura, que el cliente quiere monitorear.

Crear alerta

Este caso de uso es fundamental debido a que las alertas son las condiciones que deben cumplir los eventos para que sean registrados los logs y el cliente sea notificado. Estas alertas pueden ser de tipo bandera (que se cumplan 1 vez) o ventana (que se cumplan cierta cantidad de veces en un rango de tiempo determinado).

Un ejemplo de alerta de tipo ventana sería poder registrar cuando una IP especifica ingresa al proxy reverso una cantidad de veces determinada dentro de un rango de tiempo. Un ejemplo de alerta de tipo bandera sería poder captar cuando un usuario no consigue loguearse, dando error 401 *unauthorized*.

4.3. Modelo de dominio

Del análisis realizado, se obtuvo el modelo de dominio que se presenta en la Figura 4.2, las clases Respuesta y Hit son objetos de transferencia de datos para el control de información proveniente de Elasticsearch. Las clases GeoIP, Comunicado y Log son objetos de transferencia de datos orientados al manejo y persistencia de datos no relacionales, el resto corresponde al modelo de objetos relacionales. Para ahondar en detalle ver en documento anexos: apéndice - Modelo de dominio.

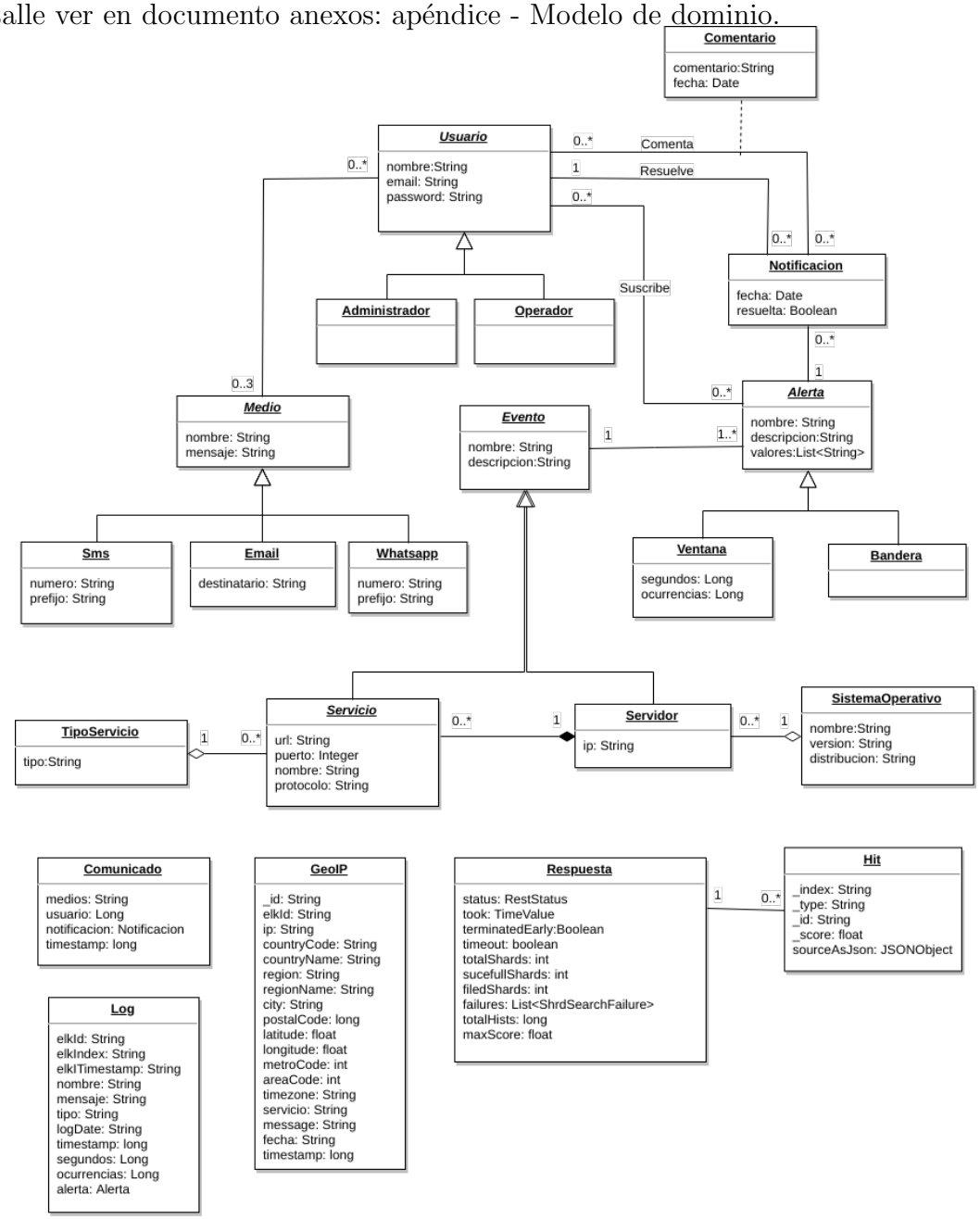


Figura 4.2: Modelo de dominio

4.4. Arquitectura del sistema

A continuación se presenta la arquitectura del sistema definido, a través de esquemas e información que detallan a grandes rasgos sus características. Provee una visión global de la estructura de los componentes de la aplicación, su interoperación y cómo se relacionan. Para ahondar en detalle ver en documento anexos: apéndice - Arquitectura.

4.4.1. Diagrama de distribución

El diagrama de distribución que se visualiza en la Figura 4.3, contiene el detalle de la arquitectura y la interacción de la aplicación.

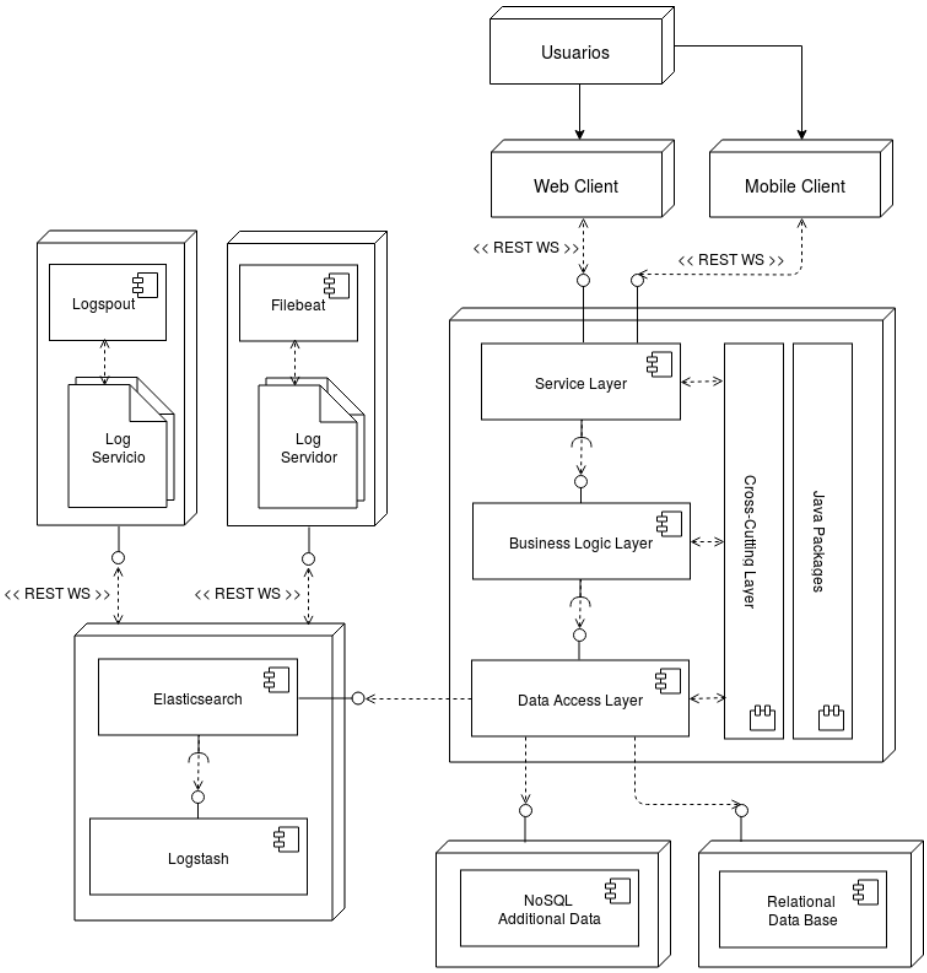


Figura 4.3: Estilo arquitectónico

4.4.2. Nodos

En el diagrama anterior se visualizan diferentes nodos, que se resumen en este apartado.

Nodo de servidor web

El nodo de servidor web tiene como propósito desplegar la aplicación cliente web de tipo *Single Page Aplication*, este nodo se comunica con la aplicación "Miralogs" utilizando servicios REST provistos a través de la capa de servicios.

Nodos de dispositivos Android

Los nodos de dispositivos Android tienen como propósito desplegar la aplicación cliente móvil de forma nativa para este sistema operativo, estos nodos se comunican con la aplicación "Miralogs" utilizando servicios REST provistos a través de la capa de servicios.

Nodo servidor de aplicaciones

El nodo servidor de aplicaciones tiene como propósito desplegar la aplicación "Miralogs" la cual centralizara toda información que disponga el cliente.

Nodo ElasticStack

El nodo ElasticStack contendrá los servicios de Logstash y Elasticsearch con el propósito de transformar y encontrar datos enviados por los nodos de infraestructura, además proveerá a la aplicación "Miralogs" con datos a través de la interfaz REST de Elasticsearch.

Nodo de base de datos no relacional

El nodo de base de datos no relacional tiene como propósito almacenar en una base orientada a documentos colecciones con información adicional al modelo relacional.

Nodo de base de datos relacional

El nodo de base de datos relacional tiene como propósito reflejar los datos a partir del modelo de objetos.

4.5. Modelo de diseño

A continuación, se muestran los diagramas de interacción para los casos de uso definidos como críticos. Para visualizar la información del resto de los casos de uso, dirigirse al documento anexos: apéndice - Modelo de Diseño.

4.5.1. Diagrama de interacción del caso de uso crítico crear evento

El evento resuelve la capacidad para considerar y clasificar datos a nivel de sistema operativo o servicios tomados desde Elasticsearch, como se muestra en la Figura 4.4.

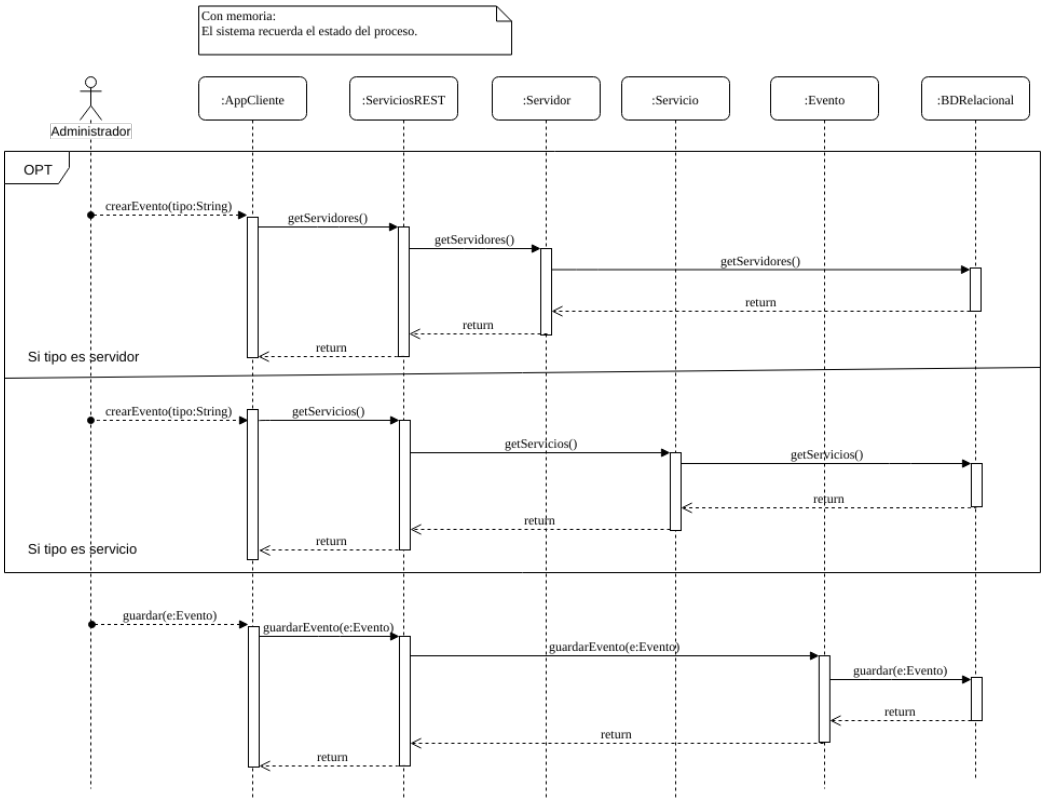


Figura 4.4: Crear evento

4.5.2. Diagrama de interacción del caso de uso crítico crear alerta

La alerta es la condición que se configura sobre un evento de tipo servidor o servicio para que genere logs y los notifique, como se muestra en la Figura 4.5. Es el análogo a un observador que espera un resultado sobre un evento.

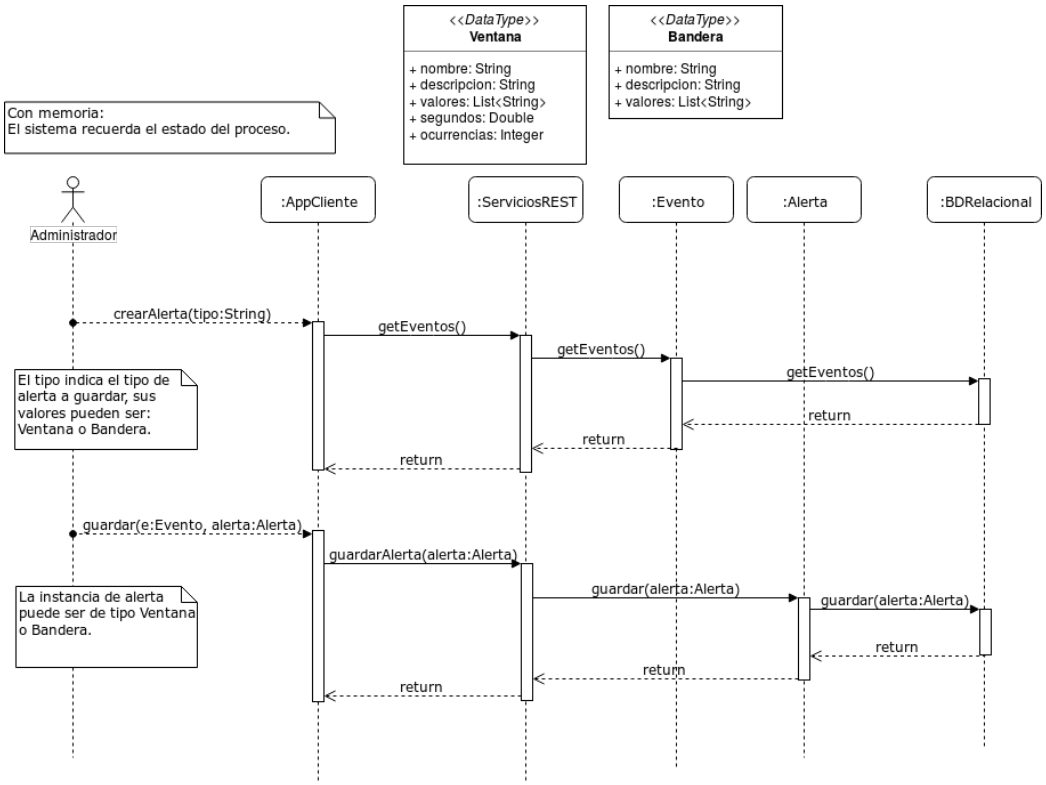


Figura 4.5: Crear alerta

Capítulo 5

Implementación

En esta sección se explica el proceso de implementación realizado para la aplicación solicitada. Para esto se tuvo especial atención en el análisis y diseño explicitados en los capítulos anteriores.

5.1. Metodología de trabajo

Con base en los conocimientos previos y las áreas de preferencia de cada integrante del equipo, se definieron los roles que se llevarían adelante en este proyecto. Éstos fueron los siguientes: Desarrollador *backend* y base de datos: persona responsable de la investigación e implementación de las tecnologías más adecuadas para la seguridad y persistencia de los datos. Desarrollador plataforma móvil: persona responsable del estudio, diseño e implementación de la aplicación Android. Desarrollador web: persona responsable del estudio, diseño e implementación de la aplicación web. Tester y documentador: persona responsable de la recabación de datos para generar la documentación final y de probar el producto en las diferentes etapas de desarrollo.

Una vez definido quién o quienes ocuparían cada rol, se realizó la planificación teniendo en cuenta el cronograma solicitado en la letra con los requerimientos. El tiempo total de esfuerzo sería 16 semanas divididas de la siguiente forma: elaboración del documento de estado del arte, análisis y diseño con entrega de prototipo, implementación y testing, documentación final y preparación de presentación. Esta planificación se realizó en la herramienta Trello [9] para registrar las tareas a realizar y para que poder hacer seguimiento constante del estado de éstas.

Para las primeras dos etapas se definieron reuniones dos veces por semana de coordinación en el grupo y quincenales con la tutora para exhibir los avances y también

recibir *feedback* de las tareas realizadas hasta el momento. Luego, en las etapas posteriores, se realizaron reuniones grupales casi todos los días de la semana, para así avanzar a la par y colaborar de primera mano entre todos para no tener inconvenientes bloqueantes.

Las reuniones se llevaron adelante de forma presencial la mayor parte del tiempo, pero cuando esto no era posible, se utilizaron medios de comunicación para que el equipo se mantenga en constante conexión. Estos medios utilizados fueron: Gmail [23], Hangouts [24] y Whatsapp [90].

5.1.1. Dedicación horaria

Se planificó un promedio de cuatro horas diarias de lunes a viernes, esto da un total de 20 horas semanales por integrante. Teniendo en cuenta que la duración del proyecto sería 16 semanas aproximadamente, para todas las etapas da como resultado 1600 horas estimadas disponibles.

El proyecto se dividió en las siguientes cuatro etapas: Investigación, Documentación, Implementación y Conclusión. Se entiende como conclusión el cierre del proyecto donde se termina documentación, se realiza el *testing*, se preparan los datos de prueba y la presentación. Cada etapa se distribuye en ciclos homogéneos de cuatro semanas. De la estimación planteada se desprende el siguiente cuadro:

Etapa	**Estimación**	**Dedicación**	**Diferencia**
Investigación	400	250	-150
Documentación	400	350	-50
Implementación	400	1000	600
Conclusión	400	300	-100
TOTALES	1600	1900	300

Cuadro 5.1: Cuadro de estimaciones.

Se deduce del Cuadro [5.1] que el proyecto llevó un poco más de 18 % de la cantidad de horas estimadas. Claramente la diferencia se presentó en la etapa de implementación, en el resto de las estimaciones sobraron horas, a causa de que faltó visión de proyección en el análisis y diseño de algunos puntos del sistema, esto se explica en la sección problemas encontrados. A continuación se representa el gráfico de la Figura 5.1 con el porcentaje de tiempo que llevó cada etapa:

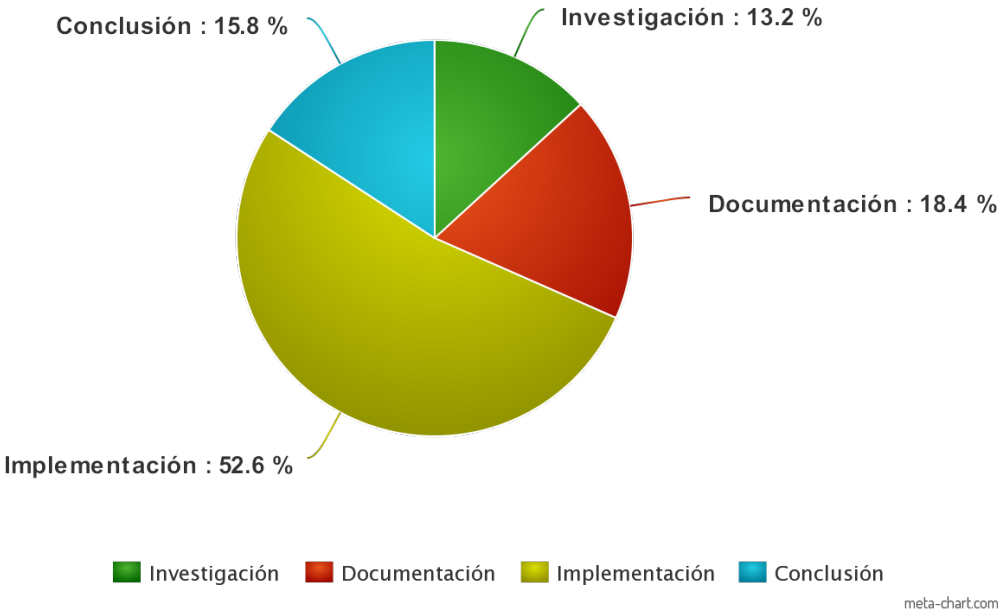


Figura 5.1: Horas reales de dedicación

5.2. Entornos de desarrollo y ejecución

Este punto presenta los entornos utilizados para el desarrollo y la ejecución de la implementación de la solución.

5.2.1. Tecnologías aplicadas

Este apartado tiene como finalidad principal, exhibir las tecnologías utilizadas en cada una de las capas de los productos. Dentro de éstas se encuentran algunas que fueron de carácter obligatorio y otras que fueron seleccionadas de acuerdo a las necesidades identificadas para alcanzar el objetivo.

Latex [36] es un sistema de composición de textos, orientado a la creación de documentos escritos que presenten una alta calidad tipográfica. Fue utilizado para hacer toda la documentación.

Docker [15] automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización a nivel de sistema operativo en Linux. Utiliza características de aislamiento de recursos del kernel Linux, tales como *cgroups* y espacios de nombres (*namespaces*) para permitir que "contenedores" independientes se ejecuten dentro de una sola instancia de Linux, evitando la sobrecarga de iniciar y mantener máquinas virtuales. Fue utilizado para tener todos el mismo ambiente de desarrollo y para facilitar el despliegue de las aplicaciones.

Capa de persistencia

Para la capa de persistencia se decidió la utilización de una base de datos relacional y otra no relacional, PostgreSQL [79] y MongoDB [46] respectivamente. PostgreSQL fue utilizado para el guardado de casi toda la información de los sistemas a excepción de los comunicados, registros de login y logs, que fueron guardados en MongoDB.

MongoDB no guarda los datos en tablas como se hace en las base de datos relacionales sino que almacena estructuras de datos en documentos similares a JSON [86] con un esquema dinámico. Para el caso de esta tecnología, se utiliza una especificación llamada BSON [45]. Esto genera que la integración de los datos en ciertas aplicaciones sea más fácil y rápida.

Para la conexión de los objetos Java con la base de datos, se utilizó Hibernate [69]. Se escogió esta tecnología debido a que hay mucha documentación y es de fácil entendimiento.

Capa de negocio

Es la capa donde se implementan los servicios REST [94] que expone la capa de servicios. Los servicios REST realizan la comunicación entre el backend y el frontend de las diferentes aplicaciones implementadas. Se optó por la opción REST en lugar de SOAP [93] ya que es una tecnología mucho más flexible y que transporta datos por medio del protocolo HTTP. Permite la utilización de los métodos que proporciona HTTP para comunicarse (GET, POST, PUT, DELETE, PATCH) y también los códigos de respuesta nativos de HTTP como por ejemplo 404 (not found) y 200 (ok). REST permite, gracias a su flexibilidad, transmitir cualquier tipo de datos. Permite el envío de XML, JSON, Binarios y otros. En el caso de este proyecto, la comunicación se realiza mediante el envío y recepción de peticiones JSON.

En el *backend* se utilizaron anotaciones en las entidades para el mapeo relacional mediante Hibernate ORM (*Object relational mapping*), para definir la estructura de datos relacional.

También en esta capa se incluyó la API de *Java High Level REST Client* [19]. Esta API nos permitió trabajar desde Java [61] con Elasticsearch [16] de forma sencilla.

Para los medios de comunicación se utilizaron las dependencias JavaMail API [55] y Twilio [81], para enviar e-mail y SMS [32] respectivamente.

Para ubicar geográficamente las peticiones por las IP que consultan al proxy reverso, se utilizó la GeoIP API [39].

Capa de servicios

En la capa de servicios se utilizaron servicios REST, que son los implementados en la capa de negocios. La capa de servicios es la responsable de exponerlos para el consumo de las diferentes aplicaciones. Se implementó la autenticación basada en Token JWT [10] con JAX-RS, esto permite un filtro de autenticación y autorización por rol, a los servicios [62].

Capa de presentación

Esta capa es fundamental ya que es la cara de la aplicación hacia el cliente. Se torna imprescindible que la misma sea atractiva y sencilla de utilizar. A su vez, la implementación en la aplicación móvil y en el sitio web deben ser consistentes en sus colores y estilos, para que de esta forma los usuarios puedan identificar con facilidad que ambas pertenecen al mismo sitio.

Web

Para el desarrollo de la aplicación web se utilizaron varias tecnologías entre las que se destacan Angular 5.2.11 [6] como *framework* JavaScript [89] para el desarrollo de aplicaciones web, que incluye TypeScript 2.5.3 [42] como parte de su set de herramientas. Bootstrap 4.0 [82] como framework de diseño incluye HTML5 [88] y JQuery 3.3.1 [77] como complemento. OpenLayers 4.1.1 [52] para el despliegue del mapa con las ubicaciones de las peticiones gestionadas por GeoIP, MomentJS 2.22.2 [44] que mejora el manejo de fechas en javascript principalmente para ChartJS 2.7.2 [12] usado en el despliegue de gráficas, para la encriptación de contraseñas en SHA512 se usa CryptoJS 1.0.7 [11] y ng2-SimpleTimer 1.3.3 [50] como timer para actualizar la información en pantalla cada treinta segundos.

Móvil

La aplicación móvil se creó utilizando React Native [66] como framework Javscript para la generacion de aplicaciones moviles nativas, además el desarrollo incluyó arquitectura Flux [21] para la creación de interfaces de usuario. Para ubicar las perticiones gestionadas por GeoIP se usó React Native Maps [67] que implementa Google Maps [25], MomentJs para faclilitar el manejo de fechas, React Native Pure Chart [51] para el despliegue de gráficas, CryptoJs para la encriptar las contrasñas en SHA5212.

5.2.2. Proceso de solución de la aplicación

Para la solución del proceso de obtención de información de logs fue necesario crear un objeto de transferencia de datos de nombre **Respuesta**, según los datos de retorno de la consulta a Elasticsearch mediante la API Java de búsqueda. La consulta se realiza cada un minuto a través de una tarea programada configurada en la clase *ElasticSchedule*. Como parámetros se selecciona el indice y la cantidad de registros a encontrar (por defecto se encuentran los últimos trecientos registros del indice especificado).

Luego de obtener la información de la consulta cargada en un objeto de tipo **Respuesta** se clasifica cada registro según su naturaleza que puede ser log de servicio, log de servidor o respuesta proxy. En caso de los tipos logs se comprueba si cumplen las alertas configuradas, si se cumple se persiste el log en la base no relacional y es notificado. En caso de respuesta proxy se obtiene la ubicación por IP y se persiste en la base de datos no relacional. Para finalizar el proceso los datos no relacionales son consultados por periodos de tiempo por las aplicaciones cliente. En la Figura 5.2 se detalla el proceso descrito.

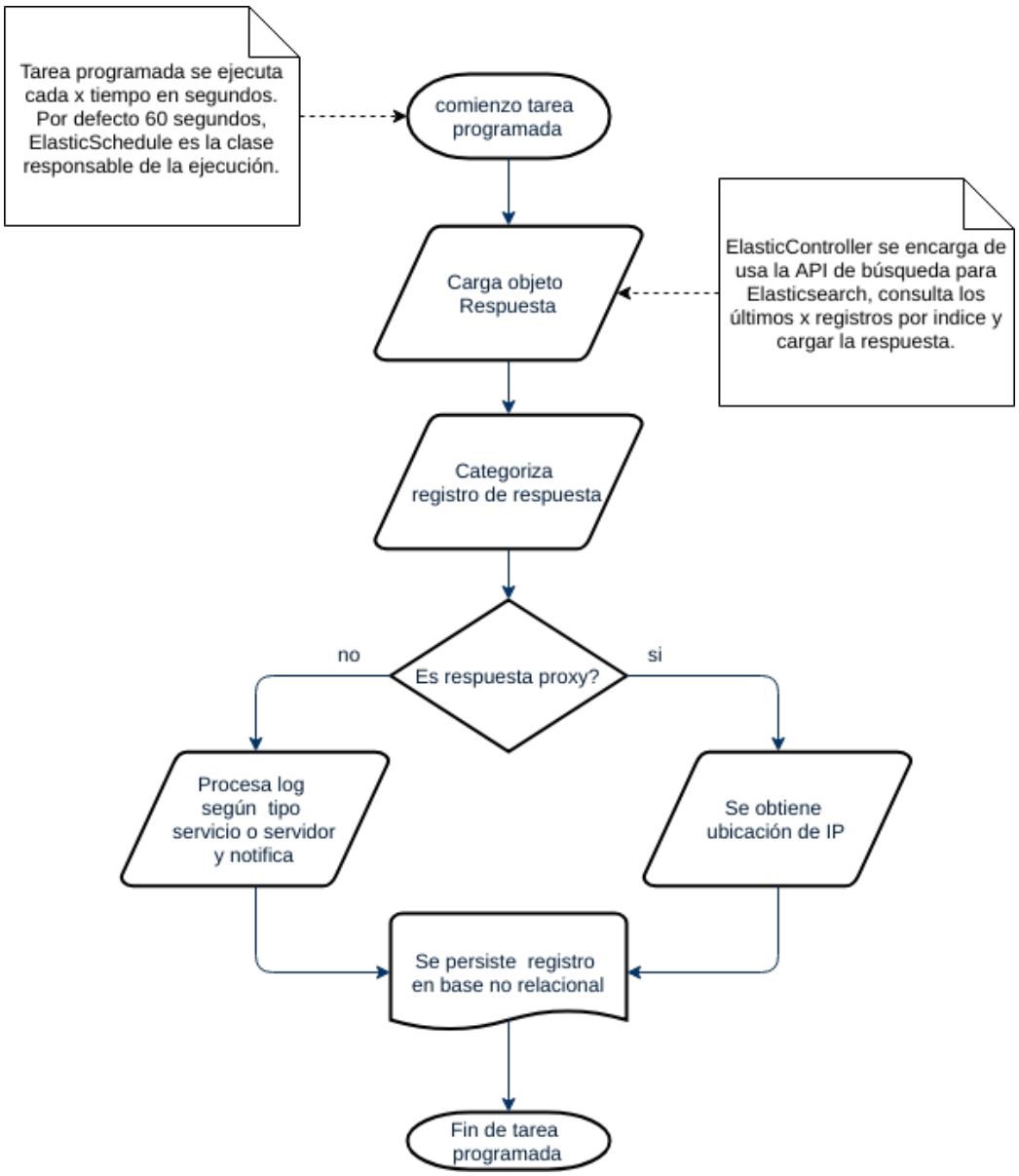


Figura 5.2: Flujo de aplicación

5.3. Aplicaciones desarrolladas

En esta sección especifica los productos de aplicación desarrollados a nivel de presentación como el cliente web y móvil.

5.3.1. Aplicación web

A continuación se describe a grandes rasgos algunas de las funcionalidades más interesantes de la aplicación web.

Dashboard

La Figura 5.3 refiere al *dashboard*, que muestra la cantidad de mensajes, comentarios, notificaciones y eventos generados. También reportes como cantidad de peticiones al proxy reverso por país y cantidad de logs generados por tipo en la ultima semana. Los datos se actualizan cada treinta segundos.

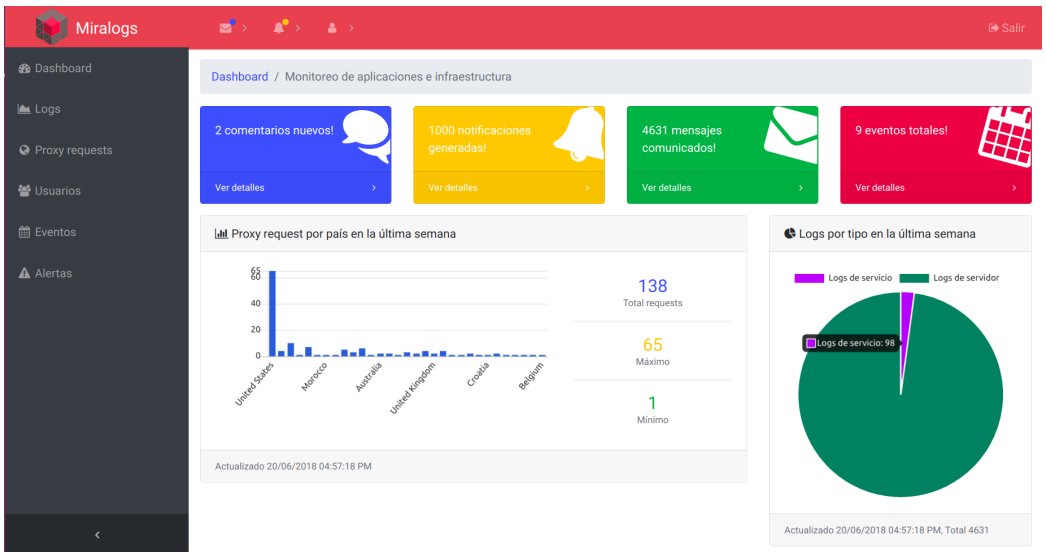


Figura 5.3: Dashboard web

Reporte de logs

El reporte de logs se puede consultar por termino de búsqueda según el atributo seleccionado. Los periodos de tiempo pueden ser relativos (por ejemplo últimos 15, 30 ó 60 minutos) o absolutos (fecha y hora fijas de inicio y fin de periodo). Los datos

se muestran en forma de tablas y en el gráfico de cantidad en función de tiempo. Todo esto se ve reflejado en la Figura 5.4.

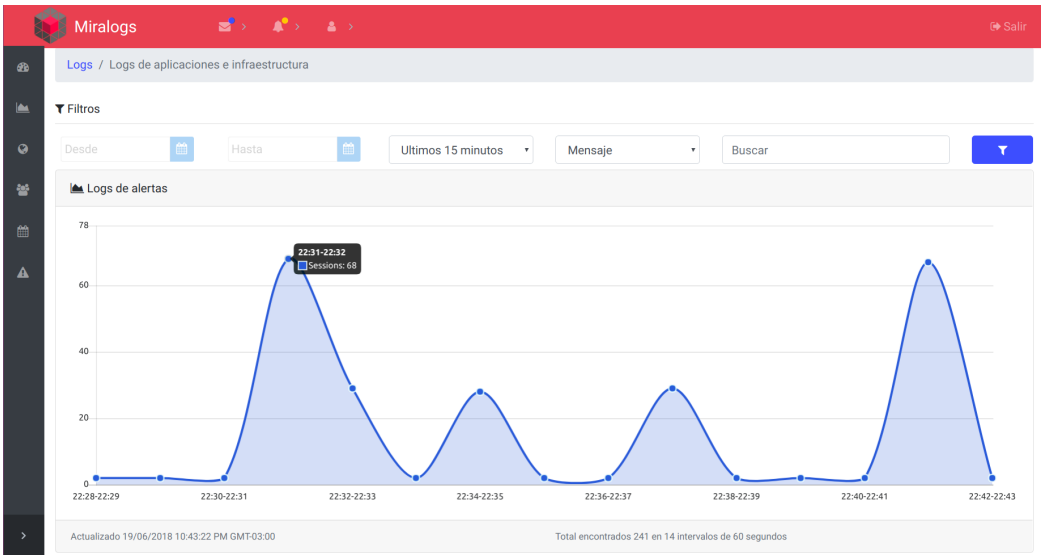


Figura 5.4: Logs web

Reporte de proxy requests

El reporte de peticiones al proxy reverso se puede consultar por termino de búsqueda según el atributo seleccionado. Los periodos de tiempo pueden ser relativos (por ejemplo últimos 15, 30 ó 60 minutos) o absolutos (fecha y hora fijas de inicio y fin de periodo). Los datos se muestran en forma de tablas y en el mapa en forma de puntos, además se pueden seleccionar peticiones en un radio de 1 a 5000 km o destacar una petición en particular, como se muestra en la Figura 5.5.

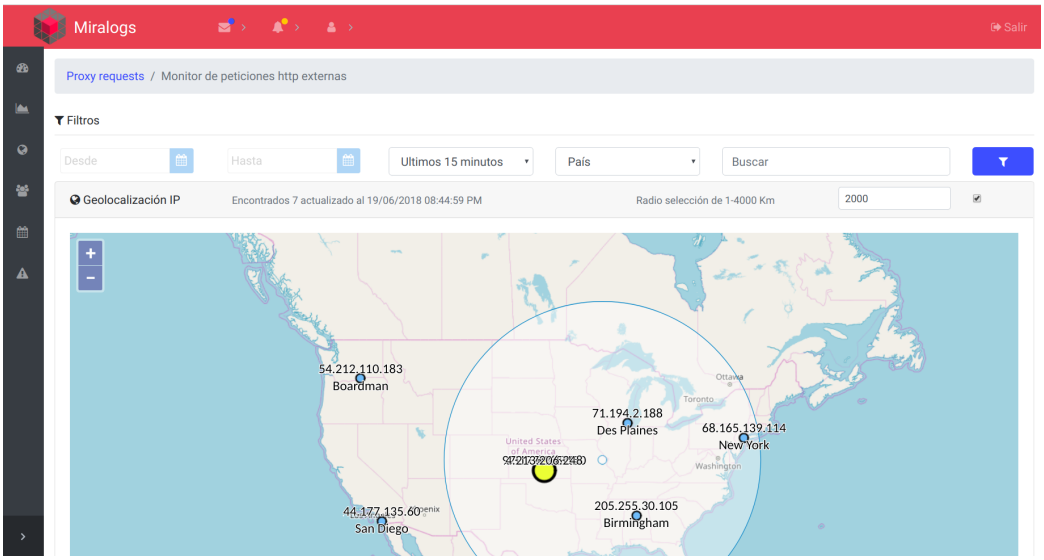


Figura 5.5: Proxy requests web

Gestión de eventos

Los eventos pueden ser de tipo servicio (logs de servicios de aplicación) o servidor (logs de servidor a nivel de sistema operativo), estos eventos se leen desde logs de Elasticsearch y se discriminan por su nombre, como se muestra en la Figura 5.6.

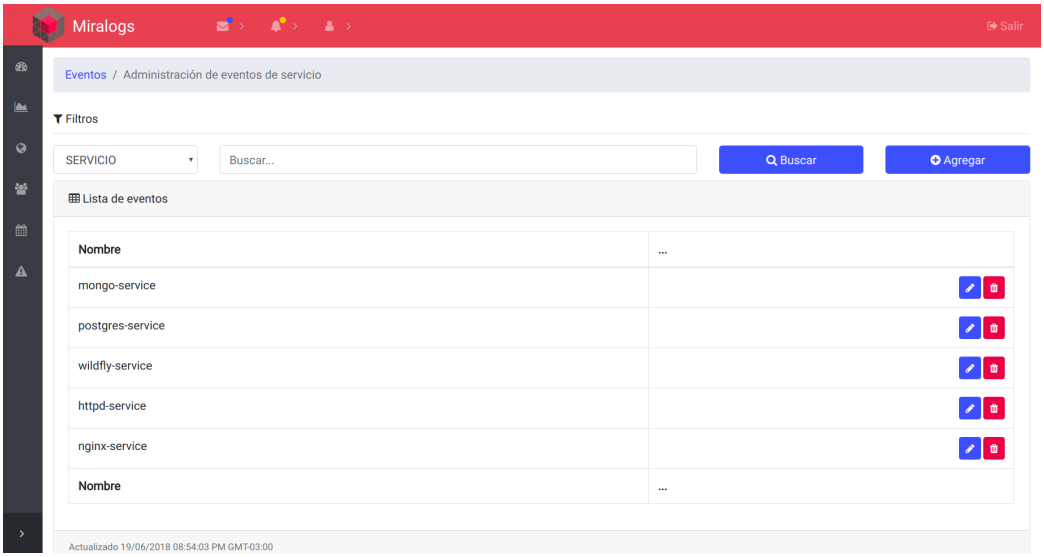


Figura 5.6: Eventos web

Gestión de alertas

En la Figura 5.7 se muestra la gestión de alertas. Las alertas son condiciones que se configuran sobre el mensaje del evento y pueden ser de tipo bandera (términos que

coincidan con el mensaje de log) o ventana (términos que coinciden con el mensaje de log n veces en x tiempo en segundos).

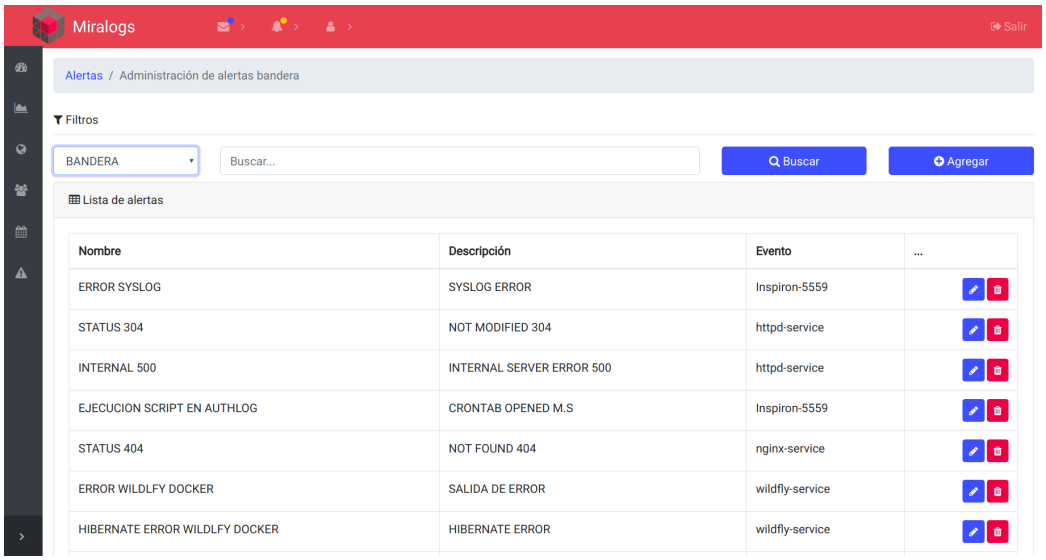


Figura 5.7: Alertas web

Gestión de usuarios

La gestión de usuarios esta disponible para los administradores y permite gestionar el resto de los usuarios, como se muestra en la Figura 5.8.

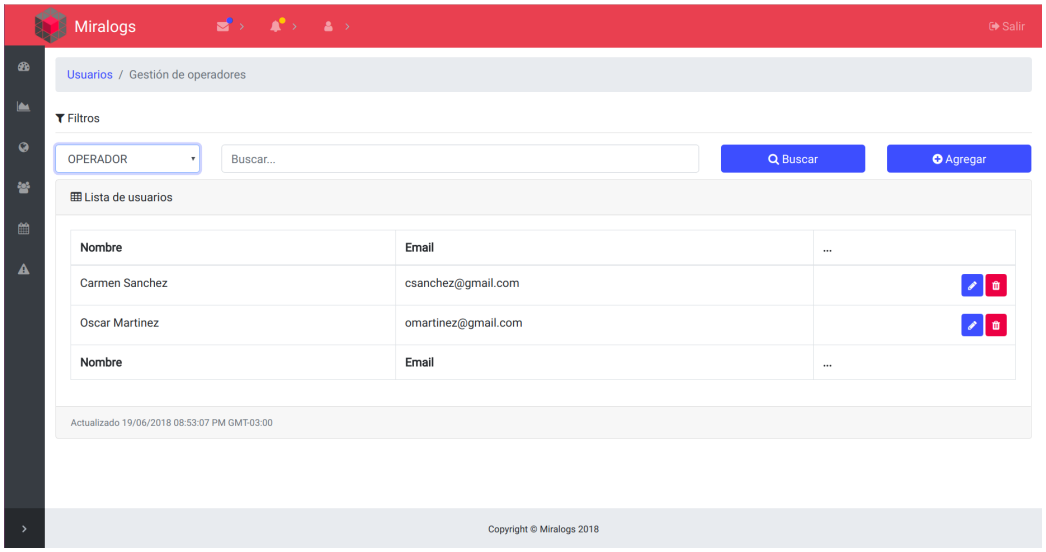


Figura 5.8: Usuarios web

Notificaciones

En la Figura 5.9 se muestran las notificaciones, que son generadas cada vez que la condición de una alerta se cumple para un evento determinado produciendo un registro de log. Por defecto las notificaciones son comunicadas a nivel de aplicación para todos los usuarios.

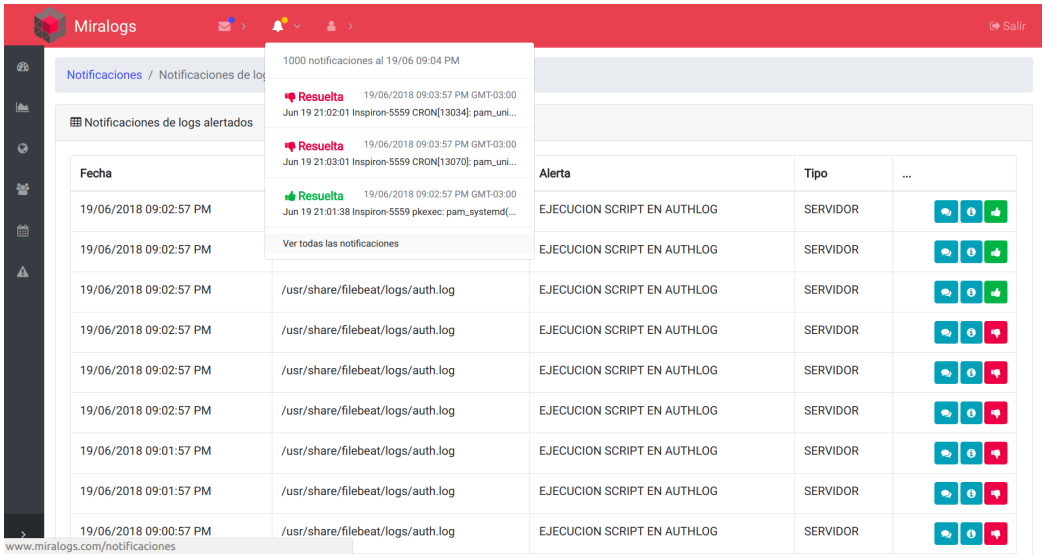


Figura 5.9: Notificaciones web

Comentarios

Las notificaciones pueden ser comentadas y/o marcadas como resueltas por cualquier usuario quedando registrado el que realizó la acción, como se muestra en la Figura 5.10.

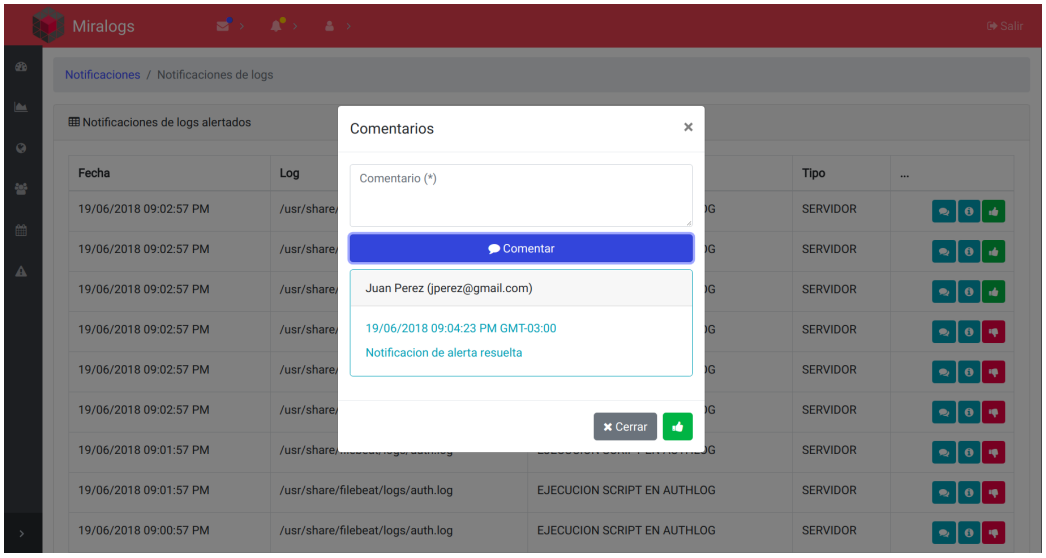


Figura 5.10: Comentarios web

Suscripción a alerta

Un usuario tiene la posibilidad de suscribirse a una alerta con el fin de recibir mensajes de las notificaciones a través de los medios de comunicación configurados como email o sms, como se muestra en la Figura 5.11.

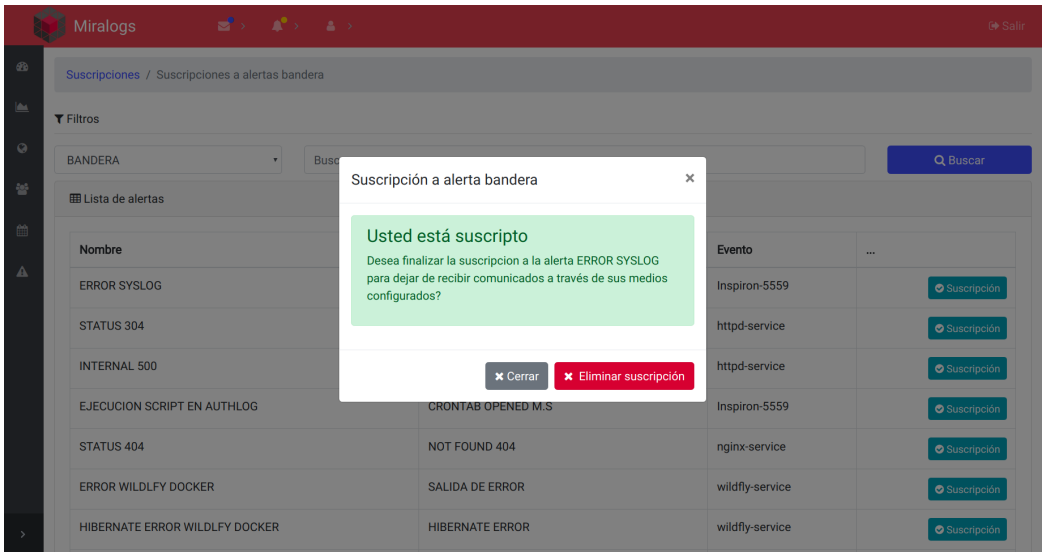


Figura 5.11: Suscripción a alerta web

Mensajes

Los mensajes son simplemente el registro de notificaciones comunicadas a través de los medios de comunicación de un usuario suscripto a una alerta, como se muestra en la Figura 5.12.

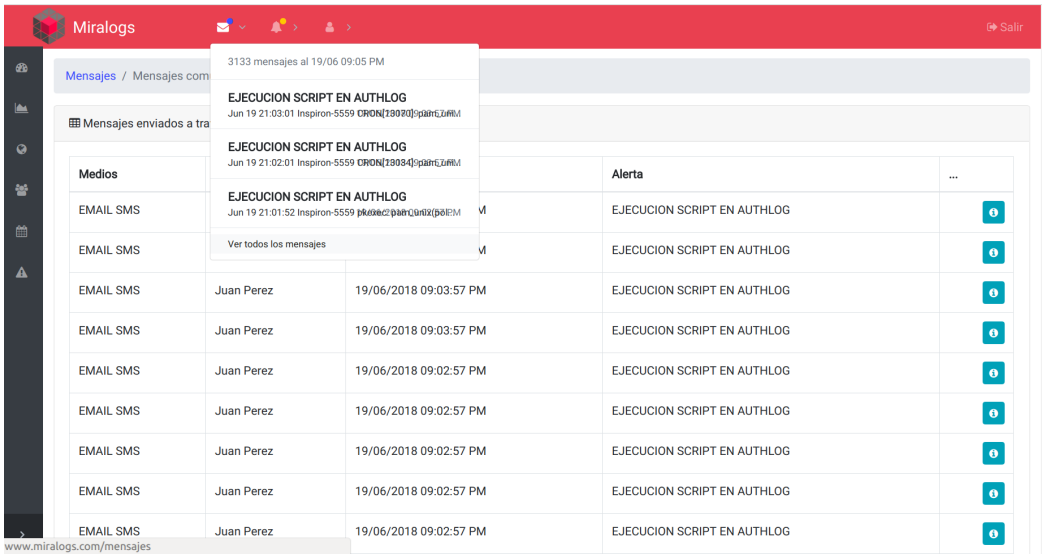


Figura 5.12: Mensajes web

5.3.2. Aplicación móvil

A continuación se describe a grandes rasgos algunas de las funcionalidades más interesantes de la aplicación móvil.

Reportes

En la Figura 5.13 se muestra el menú principal de la aplicación móvil que cuenta con configuración de cambio de idioma. Análogo a la aplicación web también cuenta con reportes de logs y proxy requests. El gráfico de logs se puede ampliar para observar a detalle, las peticiones se acceden una a una deslizando, también existen filtros por periodos de tiempo o absolutos.

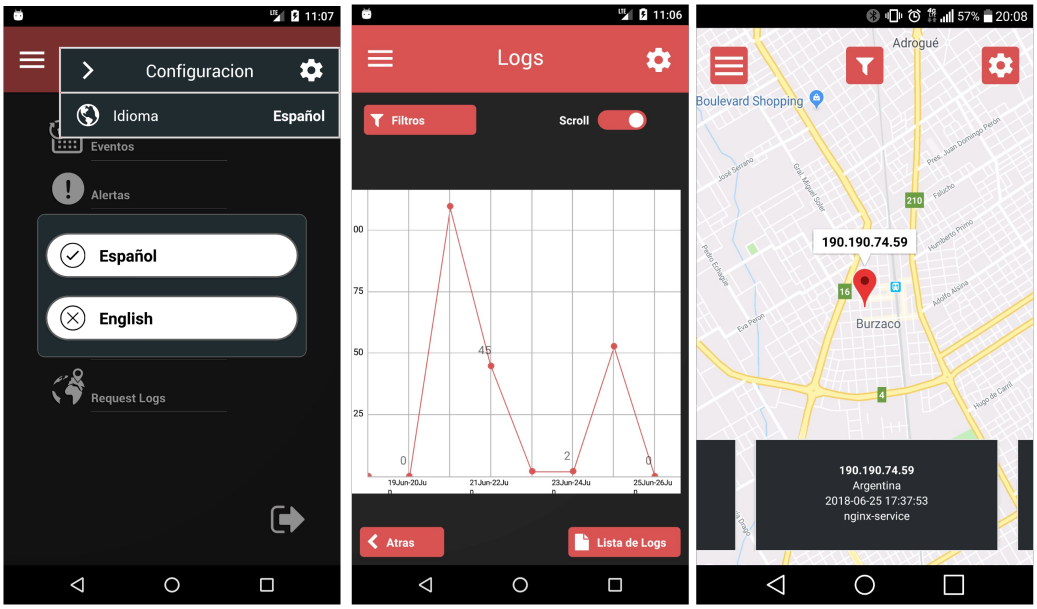


Figura 5.13: Aplicación móvil reportes

Eventos

Permite ingresar eventos de tipo servicio o servidor, listar y mostrar los detalles de cada uno según su tipo, como se muestra en la figura, como se muestra en la Figura 5.14.

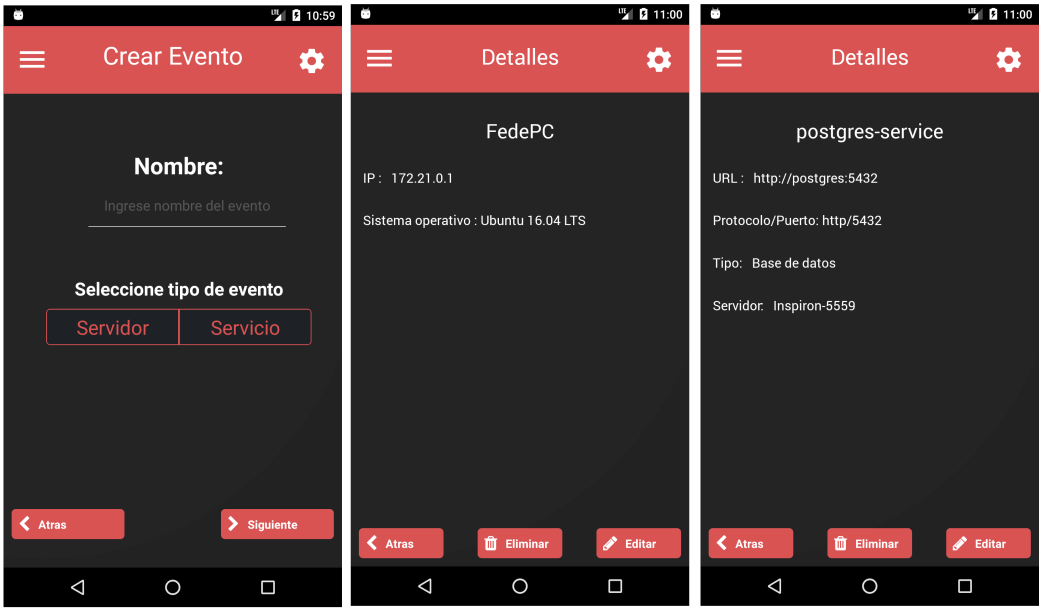


Figura 5.14: Aplicación móvil eventos

Alertas

Permite ingresar alertas de tipo bandera o ventana para un evento determinado, listar y mostrar los detalles de cada uno según su tipo, como se muestra en la Figura 5.15.

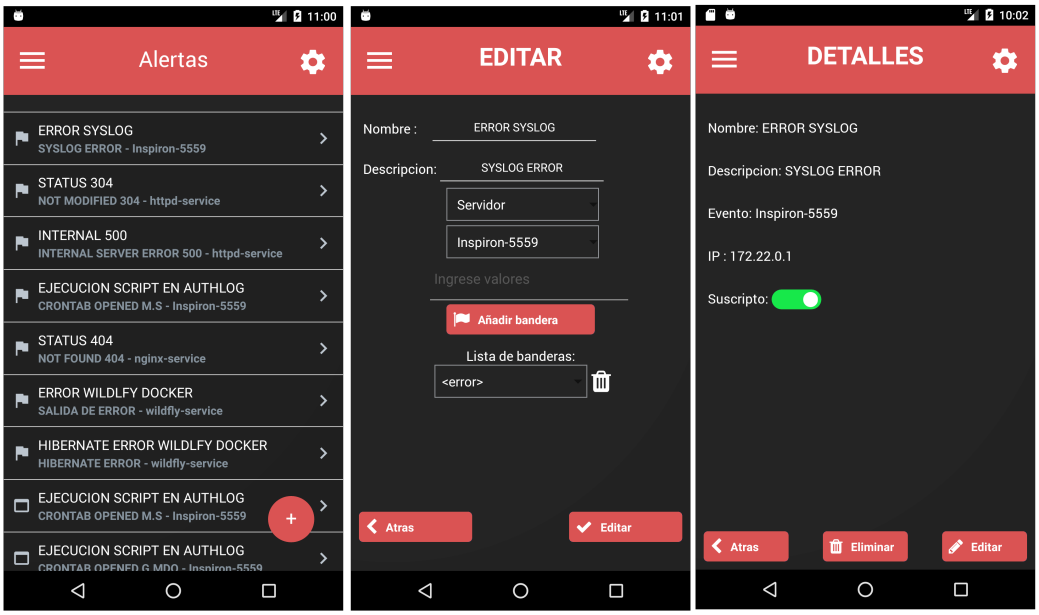


Figura 5.15: Aplicación móvil alertas

5.3.3. Problemas encontrados

Este apartado tiene como finalidad explicar los problemas enfrentados durante el desarrollo del proyecto y la forma en que el grupo se antepuso a los mismos.

Un inconveniente fue la incertidumbre en la visión del producto por falta de experiencia con Elasticsearch, si bien estaba clara su utilidad se desconocía como consumir la información y en que formato. Esto llevó a redefinir conceptos en la etapa de implementación además de que algunos casos de uso se implementaron con mucha diferencia al especificado en un principio.

Otro problema que se enfrentó, fue que los recursos del servidor de Amazon no soportaban la carga de los servicios necesarios para el funcionamiento, principalmente al añadir el stack ELK. Esto determinó que se tenga que desplegar la aplicación en un servidor local propio con una dirección de red no-ip con el objetivo de probar la aplicación móvil sin depender de IP's dinámicas en redes lan.

En un principio se había establecido React como framework de desarrollo web, pero al extenderse el tiempo de desarrollo backend se comenzó tarde a desarrollar, esto sumado a que la documentación de React no es de lo más amigable para comenzar, por lo tanto rápidamente se tomo la determinación de cambiar a Angular ya que se tenia cierto dominio previo.

Capítulo 6

Conclusiones y trabajos a futuro

6.1. Conclusiones

Se destaca el aprendizaje en herramientas como Elasticsearch, Docker, GeoIP, frameworks como React Native y Angular que impulsaron hacia la solución del problema de forma efectiva y elegante, además gracias a su utilización no hubo que preocuparse de problemas que consumen mucho tiempo, como la instalación de *software* e incompatibilidades de funcionamiento o la forma de obtener logs de aplicaciones mediante métodos mas clásicos, como recurrir a *scripts* y herramientas a nivel de sistema operativo que hubiesen hecho que el producto final sea de menor calidad o más laborioso.

El resultado obtenido es una aplicación capaz de configurar alertas sobre los mensajes de cualquier servicio o logs de sistema operativo, estas alertas pueden ser notificadas via mail o sms además de captar todas las peticiones al proxy reverso y ubicarlas geográficamente. Las funcionalidades del sistema construido aportan control sobre los servicios e infraestructura, la detección de eventos excepcionales facilita la corrección de *bugs*, aumenta seguridad y permite reaccionar rápidamente a caídas de servicios, fomentando las soluciones preventivas ante las reactivas. Trabajar de forma preventiva mejora la calidad del servicio en varios aspectos, principalmente la disponibilidad y robustez.

El equipo trabajó de forma ordenada y efectiva en cada etapa del proceso de desarrollo. El producto final y el cumplimiento de los objetivos en fecha es una clara evidencia de la comunicación continua e intercambio de opiniones. Los conflictos e inconvenientes que surgieron fueron aclarados de forma inmediata tratando de aceptar y enfocarse en el propósito común. Los roles y reparto de tareas se hicieron de acuerdo a las habilidades y comodidad de cada uno para su desempeño. El hecho de haber compartido la carrera juntos ayudó a reafirmar el grupo humano y a consolidar el compromiso.

6.2. Trabajo a futuro

Como trabajo a futuro quedaría integrar que el usuario pueda gestionar sus medios de comunicación para el envío de los mensajes notificados.

Las consultas de notificaciones, mensajes y comentarios deberían estar limitadas por fecha de forma que traiga los registros de la ultima semana o día, en el producto actual trae todos ordenados por fecha sin limitaciones, esto puede dar problemas, a medida de que se acumulen registros esas consultas van a requerir más tiempo de procesamiento a causa de la volumen de datos que se mueven.

También se pueden incluir más reportes en el *dashboard* de diferentes estadisticas sobre logs o el armado de reportes dinámicos a necesidad del usuario. Si bien la aplicación comunica las notificaciones de las alertas a las cuales se suscribió el usuario, faltaría agregar un medio alternativo como Slack o Whatsapp.

En la aplicación Android quedó pendiente la implementación de notificaciones PUSH, esto se dejó para el final ya que se contaba con notificaciones por SMS, pero por temas de tiempo no se pudo concluir.

Bibliografía

[1] ABC. API. http://www.abc.es/tecnologia/consultorio/20150216/abci--201502132105.html, 2015.

[2] Android. Android Debug Bridge. https://developer.android.com/studio/command-line/adb, 2010.

[3] Android. Android Studio. https://developer.android.com/studio/intro/, 2010.

[4] Android. About Android SDK. https://stuff.mit.edu/afs/sipb/project/android/docs/about/index.html, 2018.

[5] Android. Android. https://www.android.com/, 2018.

[6] Angular. Angular Framework. https://angular.io/, Abril 2018.

[7] Apache. Apache http server proyect. https://httpd.apache.org/, 2018.

[8] Apple. iOS. https://www.apple.com/la/ios/ios-11/, 2018.

[9] Atlassian. Trello. https://trello.com/, 2018.

[10] AuthO. JSON Web Tokens. https://jwt.io/, 2018.

[11] Brix. CryptoJs. https://github.com/brix/crypto-js, 2018.

[12] ChartJS. ChartJs. https://www.chartjs.org/, 2018.

[13] Cisco. Understanding Simple Network Management Protocol (SNMP). https://www.cisco.com/c/en/us/support/docs/ip/simple-network-management-protocol-snmp/7244-snmp-trap.html, 2018.

[14] Cisco. What Is a Firewall? https://www.cisco.com/c/en/us/products/security/firewalls/what-is-a-firewall.html, 2018.

[15] Docker Inc. What is a container. https://www.docker.com/what-container, Abril 2018.

[16] Elasticsearch. Elasticsearch. https://www.elastic.co, 2018.

[17] Elasticsearch. ELK. https://www.elastic.co/elk-stack, 2018.

[18] Elasticsearch. Filebeat. https://www.elastic.co/products/beats/filebeat, 2018.

[19] Elasticsearch. Java High Level REST Client. https://www.elastic.co/guide/en/elasticsearch/client/java-rest/master/java-rest-high.html, 2018.

[20] Elasticsearch. Logstash. https://www.elastic.co/products/logstash, 2018.

[21] Facebook. Application architecture for building user interfaces. https://facebook.github.io/flux/, 2018.

[22] Free Software Foundation. General Public License. https://www.gnu.org/licenses/gpl-3.0.en.html, 2007.

[23] Google. About Gmail. https://www.google.com/intl/es-419/gmail/about/, 2018.

[24] Google. Get started with Hangouts. https://support.google.com/hangouts/answer/2944865?co=GENIE.Platform%3DDesktop&hl=en, 2018.

[25] Google. Getting started with Google Maps SDK. https://developers.google.com/maps/documentation/android-sdk/start, 2018.

[26] Gorry Fairhurst. Transmission Control Protocol (TCP). http://www.erg.abdn.ac.uk/users/gorry/course/inet-pages/tcp.html, 2003.

[27] HoshiCorp. Introduction to Vagrant, Vagrant vs Docker. https://www.vagrantup.com/intro/vs/docker.html, Abril 2018.

[28] IBM. AIX. https://www-03.ibm.com/systems/es/power/software/aix/, 2018.

[29] IETF. CSV format specification. https://tools.ietf.org/html/rfc4180, 2018.

[30] Inovex GmbH. Zax. http://inovex.github.io/zax/, 2014.

[31] Internet Engineering Task Force (IETF). The Transport Layer Security (TLS) Protocol Version 1.2. https://tools.ietf.org/html/rfc5246, Abril 2008.

[32] Internet Engineering Task Force (IETF). URI Scheme for Global System for Mobile Communications (GSM) Short Message Service (SMS). https://www.ietf.org/rfc/rfc5724.txt, Jinio 2010.

[33] Internet Engineering Task Force (IETF). The Secure Sockets Layer (SSL) Protocol Version 3.0. https://tools.ietf.org/html/rfc6101, Abril 2011.

[34] Ionic. Ionic Framework. https://ionicframework.com/, Abril 2018.

[35] Kibana. Kibana. https://www.elastic.co/products/kibana, 2018.

[36] LaTeX. LaTeX the product. https://www.latex-project.org/, 2018.

[37] LINFO. BSD License Definition. http://www.linfo.org/bsdlicense.html, 2018.

[38] Materialize. Materialize framework. http://materializecss.com/, 2018.

[39] Maxmind. GeoIP Java API. https://github.com/maxmind/geoip-api-java, 2018.

[40] Microsoft. Active Directory. https://support.microsoft.com/es-es/help/196464, 2000.

[41] Microsoft. Exchange. https://products.office.com/es/exchange/email, 2000.

[42] Microsoft. Typescript Docs. https://www.typescriptlang.org/docs/home.html, 2018.

[43] Microsoft. Windows. https://www.microsoft.com/es-es/windows/, 2018.

[44] MomentJS. MomentJs. https://momentjs.com/, 2018.

[45] MongoDB. BSON types. https://docs.mongodb.com/manual/reference/bson-types/, 2018.

[46] MongoDB. MongoDB Docs. https://docs.mongodb.com/, 2018.

[47] Nagios. About Nagios. https://www.nagios.org/about/, 2018.

[48] New Relic, Inc. New Relic. https://newrelic.com/, 2018.

[49] Nginx. Nginx. http://nginx.org/, 2018.

[50] npm. Angular Simple timer. https://www.npmjs.com/package/ng2-simple-timer, 2018.

[51] Oksktank. React Native Pure Chart. https://github.com/oksktank/react-native-pure-chart, 2018.

[52] OpenLayers. Open Layers Docs. http://openlayers.org/en/latest/doc/, 2018.

[53] Oracle. Proxy Reverso. https://docs.oracle.com/cd/E19146-01/820-5658/gduto/index.html, 2010.

[54] Oracle. Java EE at a Glance. http://www.oracle.com/technetwork/java/javaee/overview/index.html, 2018.

[55] Oracle. Java Mail API. http://www.oracle.com/technetwork/java/javamail/index.html, 2018.

[56] Oracle. JVM. https://www.java.com/es/download/, 2018.

[57] Oracle. MySQL. https://www.mysql.com/, 2018.

[58] Oracle. Oracle. https://www.oracle.com/index.html, 2018.

[59] Oracle. Solaris. https://www.oracle.com/lad/solaris/solaris11/index.html, 2018.

[60] Oracle. Welcome to VirtualBox.org! https://www.virtualbox.org/, 2018.

[61] Oracle. What is a java, ¿Por qué debería actualizarme a la versión más reciente de Java? https://www.java.com/es/download/faq/whatis_java.xml, Abril 2018.

[62] Poysa. Best practices for rest token based authentication. https://github.com/Posya/wiki/blob/master/best-practice-for-rest-token-based-authentication-with-jax-rs-and-jersey.adoc, 2018.

[63] Rainer Gerhards. Syslog. https://tools.ietf.org/html/rfc5424, 2009.

[64] Rainer Gerhards and Adiscon GmbH. RELP. https://www.rsyslog.com/doc/relp.html, 2008.

[65] React. React Framework. https://reactjs.org/, Abril 2018.

[66] React. React Native Framework. https://facebook.github.io/react-native/, Abril 2018.

[67] React Community. React Native Map components for iOS + Android. https://github.com/react-community/react-native-maps, 2018.

[68] RedHat. Getting Started with WildFly 12, Requeriments. http://docs.wildfly.org/Getting_Started_Guide.html#requirements, Abril 2018.

[69] RedHat. Hibernate ORM. http://hibernate.org/orm/, 2018.

[70] Rsyslog. Rsyslog. https://www.rsyslog.com/, 2018.

[71] Slack. Donde tu trabajo fluye. https://slack.com/intl/es, 2018.

[72] The Apache Software Foundation. Licencia Apache. https://www.apache.org/licenses/LICENSE-2.0, 2004.

[73] The Apache Software Foundation. Apache Lucene. https://lucene.apache.org/core/, 2016.

[74] The Apache Software Foundation. Apache module API. https://httpd.apache.org/docs/2.4/mod/, 2018.

[75] The Apache Software Foundation. Apache Software Foundation. https://www.apache.org/, 2018.

[76] The Internet Society. HTTP/1.1 (RFC2616). https://tools.ietf.org/html/rfc2616/, Junio 1999.

[77] The jQuery Foundation. Jquery API Docs. https://api.jquery.com/, 2018.

[78] The Open Group 1995-2015. Unix. http://www.unix.org/what_is_unix.html, 2015.

[79] The PostgreSQL Global Development Group . PostgreSQL. https://www.postgresql.org/, 2018.

[80] TuxFamily. GNU/Linux. https://getgnulinux.org/es/linux/, 2018.

[81] Twilio. Twilio Doc SMS. https://www.twilio.com/docs/sms, 2018.

[82] Twitter. Bootstrap. https://getbootstrap.com/, 2018.

[83] Ubuntu. Ubuntu for desktop. https://www.ubuntu.com/desktop, 2018.

[84] Veeam. What is hyper-v technology? https://www.veeam.com/blog/what-is-hyper-v-technology.html, 2018.

[85] VMware, Inc. vmware software virtualization. https://www.vmware.com/, 2018.

[86] W3cSchools. Introduction to JSON. https://www.w3schools.com/js/js_json_intro.asp, 2018.

[87] w3schools. CSS. https://www.w3schools.com/css/, 2018.

[88] w3schools. HTML. https://www.w3schools.com/html/, 2018.

[89] w3schools. Javascript. https://www.w3schools.com/javascript/, 2018.

[90] WhatsApp Inc. Mensajería confiable. https://www.whatsapp.com/?l=es, 2018.

[91] World Wide Web Consortium. Adressing, Uniform Resource Locator (URL) specification. https://www.w3.org/Addressing/URL/url-spec.html, 2018.

[92] World Wide Web Consortium. Extensible Markup Language (XML). https://www.w3.org/XML/, 2018.

[93] World Wide Web Consortium. Latest SOAP versions. https://www.w3.org/TR/soap/, 2018.

[94] World Wide Web Consortium. REST, Semantic Web Standards. https://www.w3.org/2001/sw/wiki/REST, 2018.

[95] Zabbix LLC. Zabbix. https://www.zabbix.com/, 2018.