Solución de seguimiento en tiempo real al uso del repositorio de Nova

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas

Autor: Raúl Sánchez Arañó

Tutores: Ms C. Yoandy Pérez Villazón

Ing. Yoandy Arcia Brache

LA HABANA, JUNIO DE 2017

UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

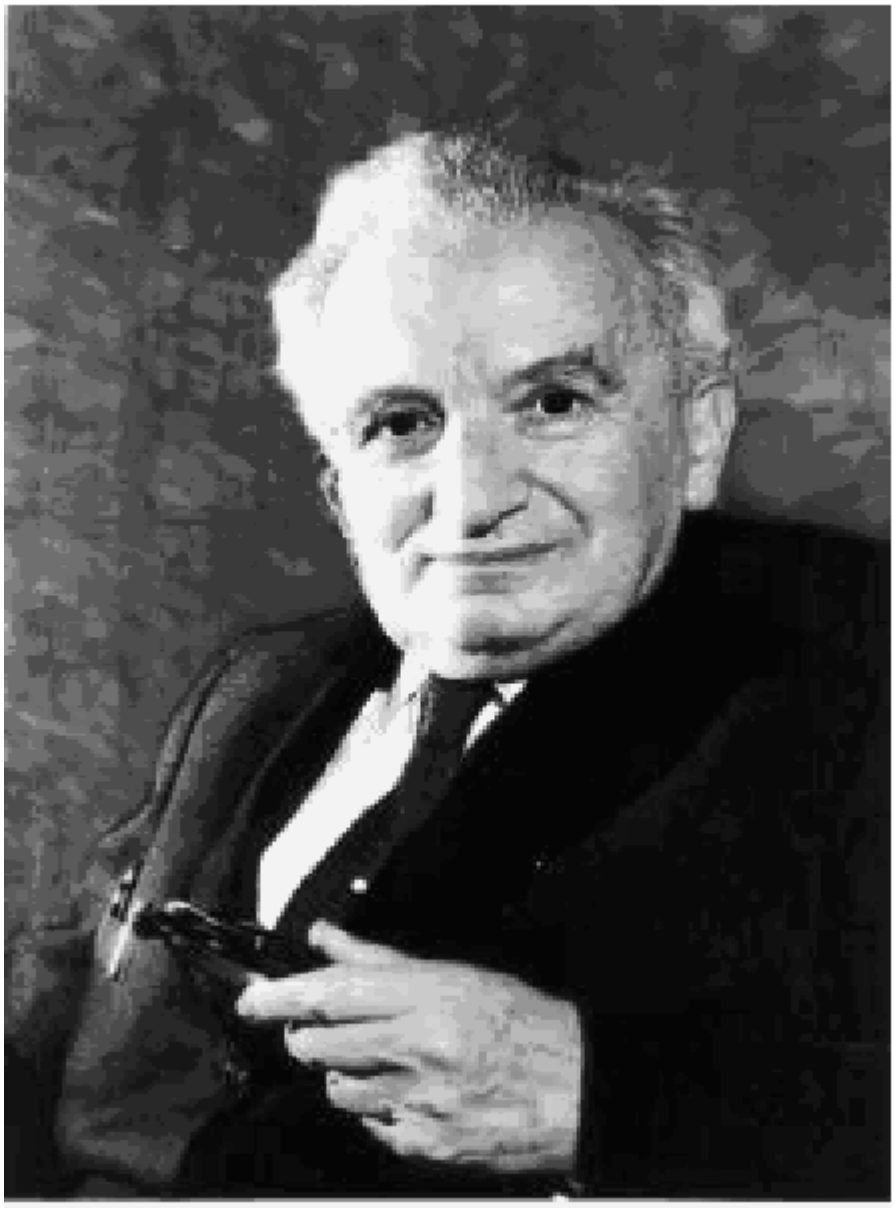
Declaración de autoría

Declaro por este medio que yo **Raúl Sánchez Arañó**, con carné de identidad **93071825066** soy el autor principal del trabajo titulado **"Solución de seguimiento en tiempo real al uso del repositorio de Nova"** y autorizo a la Universidad de las Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo. Para que así conste firmo la presente declaración jurada de autoría en La Habana a los _____ días del mes de _____ del año _____.

Firma del autor

Firma del tutor

Firma del tutor



“Los científicos exploran lo que es y los ingenieros crean lo que nunca ha sido”

Theodore Von Kármán

Dedicatoria

A mis padres, incluyendo a Sergio y a Yoly por el cariño y la preocupación que han tenido conmigo en todo momento.

A mis 4 hermanos por darme siempre esa alegría, en especial a Richard que nunca se ha separado de mí.

A mi familia en general por darme fuerza para continuar

Agradecimientos

A mis padres que gracias a ellos soy quien soy.

A Sergio y a Yoly por acogerme y ser un hijo más.

A mis hermanos por estar siempre en todo momento, y los que están lejos los tengo siempre presente.

A mis compañeros de aula y de proyecto.

A mis tutores por aconsejarme para poder lograr este resultado.

A Yanet Cabeza, que sin su ayuda y apoyo no hubiese sido posible lograr este resultado.

Resumen

GNU/Linux Nova es la distribución cubana propuesta por la Comisión de Informatización y Ciberseguridad para ser empleado en los Organismos de Administración Central del Estado (OACE). El uso de la distribución requiere que los usuarios accedan a repositorios de *software* para instalar nuevas aplicaciones y actualizar los sistemas instalados. Los administradores y desarrolladores de Nova no poseen una herramienta que brinde los datos necesarios para mejorar los servicios y ofrezca los conocimientos que posibiliten una mejor experiencia de uso del repositorio. Partiendo de esta situación se define como objetivo general de la investigación implementar un sistema informático de monitoreo en tiempo real que permita conocer las estadísticas acerca de la instalación de las aplicaciones del repositorio de Nova. Para implementar el sistema se escogieron las herramientas *Logstash* que es la encargada de analizar y procesar la información contenida en el archivo *log*, *Elasticsearch* que almacena e indexa dicha información procesada, y finalmente *Kibana* accede a los datos indexados para mostrar la información en forma de gráficas y tablas mediante los llamados *dashboards*. Los *dashboards* fueron configurados para visualizar diferentes estadísticas como: el paquete más descargado, la cantidad de paquetes descargados por IP, la cantidad de descargas por fecha, entre otros. Se realizaron las pruebas funcionales para valorar la correcta funcionalidad del sistema y mediante la aplicación de la técnica de IADOV se reflejó la aceptación de la propuesta y el reconocimiento a su utilidad.

Palabras clave: *dashboards***,** herramientas, *logs,* pruebas, repositorio, *software*.

Índice de ilustraciones

Índice de tablas

Introducción

En los últimos años, las tecnologías de la información y la comunicación (*TIC*) han tomado un papel importante en la sociedad y se utilizan en disímiles actividades. Las *TIC* forman ya parte de la mayoría de sectores: educación, robótica, administración pública, empleo y empresas. Se desarrollan a partir de los avances científicos producidos en los ámbitos de la informática y las telecomunicaciones. Son el conjunto de tecnologías que permiten el acceso, producción, tratamiento y comunicación de información presentada en diferentes códigos (texto, imagen, sonido y video).

Cuba no ha estado exenta del uso de las tecnologías, desde la década de los 80 del pasado siglo se han empleado las tecnologías de la información. El uso de la informática estuvo soportado sobre plataformas privativas adoptando el sistema operativo *Microsoft Windows*. En la actualidad predomina la percepción por parte de las autoridades del país de que no puede apostarse por estos sistemas operativos como una solución a largo plazo para Cuba. Uno de los motivos más fuertes que impulsa a la isla para llevar a cabo un proceso migratorio hacia *software* libre lo constituye el tema de la soberanía tecnológica.

El departamento de Sistema Operativo del Centro de *Software* Libre (CESOL) de la Universidad de las Ciencias Informáticas (UCI) es el encargado del desarrollo de la Distribución Cubana de GNU/Linux Nova, que es desarrollada en apoyo a la migración a tecnologías de *Software* Libre y Código Abierto en Cuba. El proyecto Nova provee una línea de productos y servicios de calidad orientado a usuarios nacionales y extranjeros que se desempeñan en el área de las tecnologías de *software* libre.

El país actualmente se encuentra inmerso en este proceso de cambio en las tecnologías informáticas; se ha venido realizando un estudio que pueda llevar a cabo una migración de los principales *softwares* y sistemas operativos a sus posibles alternativas del *software* libre. En este punto es donde entra a tomar parte importante la Distribución Cubana de GNU/Linux Nova como alternativa de migración. Nova es la distribución propuesta por la comisión de informatización y ciberseguridad para ser empleada en los OACE, y en las computadoras se requiere el acceso a repositorios de *software* para instalar nuevas aplicaciones y actualizaciones.

A partir del año 2010 la UCI estableció una alianza estratégica de trabajo con la empresa GEDEME con el objetivo de instalar Nova de forma predeterminada en todas las computadoras ensambladas por esta empresa, las cuales tienen como destino mayoritario los OACE. Las computadoras al salir del GEDEME

poseen un sistema operativo con las aplicaciones básicas de propósito general, como por ejemplo Libre *Office* y *Firefox*.

Cada vez que se instala o actualiza alguna aplicación, el repositorio de aplicaciones emite un *log* donde se refleja el tiempo, aplicaciones que se instalan y dirección *IP*[1]. Debido a la cantidad de computadoras que se conectan al repositorio, el volumen de datos generado es elevado. El análisis de las aplicaciones que los usuarios instalan, así como las que no pueden instalar por errores del repositorio es un dato de gran utilidad para el desarrollo de la distribución Nova, pues permite priorizar el desarrollo de aquellas aplicaciones que son más usadas por los usuarios, posibilita enfocar los esfuerzos en la mejora de dichas herramientas y brinda un medio eficaz para conocer cuáles son las herramientas que presentan errores en el repositorio.

Actualmente, para conocer las aplicaciones más descargadas por los usuarios, los desarrolladores de Nova realizan un análisis de los ficheros de *logs* del servidor de aplicaciones que publica el repositorio de forma manual empleando expresiones regulares en la búsqueda de determinados patrones de interés, lo cual es una tarea muy engorrosa por el cúmulo de información almacenada.

A partir de lo anteriormente planteado, se identifica el siguiente **problema científico:** ¿Cómo monitorear en tiempo real el uso del repositorio de Nova para conocer las diferentes estadísticas que se pueden extraer a partir de los *logs* generados por el repositorio?

El **objeto de estudio** en el cual se enmarca el problema planteado, lo constituyen las herramientas para recopilar y procesar la información acerca del uso de los repositorios de distribuciones libres.

La investigación tiene como **objetivo general** implementar un sistema informático de monitoreo en tiempo real que permita conocer las estadísticas acerca de la instalación de las aplicaciones del repositorio de Nova.

En el marco de la investigación el objetivo general se ha desglosado en los siguientes **objetivos específicos:**

- Estudiar las herramientas y soluciones existentes para el análisis de registros de información.

[1] Es un acrónimo para *Internet Protocol*, son un número único e irrepetible con el cual se identifica una computadora conectada a una red

- Diseñar un sistema informático que monitoree el uso del repositorio de Nova y ofrezca su análisis y las estadísticas correspondientes.
- Valorar la solución propuesta, a partir de la simulación de varios entornos de pruebas y mediante la aplicación de un método de consulta a expertos.

Para cumplir con los objetivos anteriormente expuestos se definen las siguientes **tareas de investigación:**

- Fundamentación teórica sobre las herramientas diseñadas para el análisis de los repositorios.
- Estudio de la estructura y funcionamiento de los registros del repositorio de Nova.
- Implementación de un sistema informático que monitoree en tiempo real el uso del repositorio de Nova.
- Diseño y ejecución de las pruebas al sistema de recopilación de información de los repositorios de Nova.

Durante el proceso de investigación se definieron las siguientes **preguntas científicas:**

- ¿Cuáles son los presupuestos teóricos que fundamentan el proceso de monitoreo en tiempo real al uso del repositorio de Nova?
- ¿Cuáles son las herramientas y tecnologías más adecuadas para implementar una solución de seguimiento en tiempo real del uso del repositorio de Nova?
- ¿Qué técnicas y pruebas aplicar para la validación de la solución en tiempo real al uso del repositorio de Nova?

Los métodos científicos utilizados en el proceso investigativo fueron:

• El método **Analítico-Sintético** posibilita el análisis de fuentes relevantes relacionadas con las herramientas para recopilar y procesar información acerca del uso de los repositorios y a partir de su estudio definir las características fundamentales para la realización del sistema.

• El método **Histórico-Lógico** permite analizar la evolución de las tecnologías de recopilación y

procesamiento de datos en aras de emplearla en el sistema que se pretende elaborar.

El documento se encuentra estructurado de la siguiente manera:

Capítulo 1: Fundamentación teórica.

Se muestra la fundamentación teórica acerca de las herramientas para recopilar, analizar y procesar información sobre el uso en tiempo real del repositorio de Nova. Se presentarán algunos conceptos fundamentales y características de las mismas que puedan enriquecer el desarrollo de la investigación e implementación de las diferentes tecnologías. Con un proceso de *software* guiado por la metodología de desarrollo de *software* AUP-UCI.

Capítulo 2: Descripción de la solución propuesta.

En este apartado se analiza y diseña la propuesta de solución del sistema a implementar. Además, se describen las características que debe cumplir el sistema, definiéndolas a través de los requisitos funcionales y no funcionales.

Capítulo 3: Implementación y pruebas.

Se evidencia la elaboración del sistema informático de monitoreo en tiempo real al uso del repositorio de Nova y la realización de las pruebas con el objetivo de demostrar el funcionamiento correcto de las tecnologías empleadas.

Capítulo 1. Fundamentación teórica

Introducción

En el presente capítulo se analizan los aspectos fundamentales acerca de la estructura y funcionamiento del repositorio de Nova. Además, se realiza un estudio de los conceptos asociados a la investigación, así como las herramientas de análisis del uso de los repositorios. Finalmente se establece una conclusión acerca de los resultados obtenidos de la investigación que implicará a la solución.

1.1 Conceptos necesarios de la investigación

Para lograr una mejor comprensión de la investigación, se abordan un conjunto de aspectos necesarios que están estrechamente relacionados con el dominio del problema.

Sistema informático

El autor James A. Senn en el libro de Análisis y Diseño de Sistemas de Información señala que la definición de Sistemas de Información "*Es el proceso de examinar la situación de una empresa con el propósito de mejorarla con métodos y procedimientos más adecuados*".

En el libro de Ingeniería del *Software*, el autor Ian Sommerville define que sistemas es "*la colección de componentes interrelacionados que trabajan conjuntamente para cumplir algún objetivo*".

Así mismo Kendall & Kendall en su obra Análisis y Diseño de Sistemas menciona que "*Es una colección de subsistemas interrelacionados e interdependientes que trabajan de manera conjunta para llevar a cabo metas y objetivos predeterminados*".

Para el desarrollo del presente trabajo de diploma se define como **sistema informático** al conjunto de partes interrelacionadas: *hardware*, *software* y personal informático; que permiten almacenar y procesar información con un fin determinado.

***Software* libre**

El *software* libre respeta la libertad de los usuarios y la comunidad. A grandes rasgos, significa que los usuarios tienen la libertad de ejecutar, copiar, distribuir, estudiar, modificar y mejorar el *software* (*Free Software Foundation*, 2016).

Un programa es *software* libre para el usuario y la comunidad siempre que se tenga (Stallman, 2004):

0. La libertad de ejecutar el programa sea cual sea el propósito.
1. La libertad para modificar el programa para ajustarlo a diferentes necesidades. (Para que se trate de una libertad efectiva en la práctica, se deberá tener acceso al código fuente, dado que sin él la tarea de incorporar cambios en un programa es extremadamente difícil).
2. La libertad de redistribuir copias, ya sea de forma gratuita o a cambio del pago de un precio.
3. La libertad de distribuir versiones modificadas del programa, de tal forma que la comunidad pueda aprovechar las mejoras introducidas.

GNU/Linux Nova

Las distribuciones GNU/Linux son una variante de sistemas operativos de propósito general surgidos como parte del desarrollo del *software* libre (SWL) en su búsqueda por un entorno de desarrollo donde se puedan compartir los conocimientos. En Cuba, el despliegue de SWL, ha sido una solución para la creciente industria de desarrollo de *software*, la cual se estaba viendo afectada en el pago de licencias para la producción y comercialización de *software* sobre plataformas privativas. Por esta y otras razones, como la soberanía tecnológica, Cuba se encuentra hoy en un proceso de migración hacia plataformas de SWL y código abierto. (Albo Castro, 2016)

GNU/Linux Nova es una distribución cubana desarrollada en la Universidad de las Ciencias Informáticas, la cual cuenta actualmente con 3 variantes orientadas a diferentes escenarios: Nova Ligero, Nova Escritorio y Nova Servidor.

Nova Ligero

Ligero es la edición de Nova destinada a computadoras de bajas prestaciones de *hardware*, trazándose como objetivos lograr un sistema ligero en cuanto a consumo de memoria RAM, usabilidad y navegabilidad en el entorno de escritorio Guano. Ofrece un diseño cómodo para usuarios de *Windows* y el establecimiento de una identidad propia del sistema operativo que permita diferenciarlo. Esta versión ha sido probada satisfactoriamente en ordenadores con 256 MB de RAM. (Nova, 2015)

Nova Escritorio

Es la edición estándar de Nova dedicada a los ordenadores personales. Esta variante está pensada para ordenadores con buenas prestaciones, sin embargo, tiene como requerimientos mínimos para la

instalación 1GB de RAM, y se recomienda 2 GB de RAM en el ordenador para un funcionamiento más fluido. (Nova, 2015)

Nova Servidor

Es utilizado en servidores de datos, correo, aplicaciones, proxy, ftp, bases de datos, entre otros servicios telemáticos. (Nova, 2015)

Repositorio digital

Son sistemas de información que preservan y organizan materiales científicos y académicos como apoyo a la investigación y el aprendizaje; y garantizan el acceso a la información. Se considera un sistema de gestión de contenidos, que administra la producción científica en formato digital. Utilizan estándares abiertos para garantizar que sus contenidos sean accesibles y puedan ser buscados y recuperados para su uso posterior. Un repositorio permite importar, identificar, almacenar, preservar, recuperar y exportar un conjunto de objetos digitales, desde un portal *web*. (R. Barton, Mary y M. Waters, Margaret, 2013)

Tipos de repositorios

- **Repositorio de software:** la variedad del servicio que ofertan depende del tipo de licencia usada:
    - **Licencia privativa:** el administrador limita o restringe las propiedades del *software*. Ejemplo: *Windows Update*.
    - **Licencia de uso libre:** ofrecen una plataforma de trabajo colaborativo y compartida de conocimiento libre sobre cualquier temática, sin ningún tipo de restricciones. Ejemplo: repositorios de *software* libre, paquetes para el sistema operativo GNU/Linux, desde plataformas como *SourceForge* o Forja de *Guadalinex*.
- **Repositorios institucionales:** desarrollado por organismos políticos, sociales y educativos como universidades e institutos o asociaciones, para depositar, usar y preservar la producción científica y académica que generan en formato digital y haciéndola accesible al público. De esta manera la institución ofrece un servicio acorde al movimiento de acceso abierto.
- **Repositorios temáticos:** creados por un grupo de investigadores o una institución, que reúnen documentos relacionados con un área temática particular. La temática suele ser social, de educación ciudadana o académica.

- **Repositorios de datos:** repositorios que almacenan, conservan y comparten los datos de las investigaciones.

Estructura del repositorio de Nova

Los repositorios de Nova quedan estructurados de la siguiente forma utilizando dos secciones: **principal** y extendido.

URL del repositorio: http://direccion_del_repositorio.uci.cu/nova/versión principal extendido

En la sección principal se encuentran los paquetes fundamentales de todas las versiones de Nova más las aplicaciones propuestas en la guía cubana y solicitadas por la comunidad. En la sección **extendido** quedaran el resto de los paquetes heredados de otras distribuciones compatibles, pero no mantenidos por el equipo de Nova.

Contenido del repositorio de Nova

El repositorio de Nova está compuesto por un gran número de paquetes donde se incluyen archivos de paquetes personales, y otros que no estén soportados oficialmente en *Debian* o *Ubuntu*. Se priorizan los paquetes desarrollados, adaptados o mantenidos por la comunidad cubana de desarrollo siempre que se disponga de las fuentes.

Paquete de *software*

Un paquete es un conjunto de ficheros relacionados con una aplicación, que contiene los objetos ejecutables, los archivos de configuración, información acerca del uso e instalación de la aplicación, todo ello agrupado en un mismo contenedor. Estos pueden ser binarios y de código fuente (Miranda Domínguez, y otros, 2014).

Paquetes binarios (Miranda Domínguez, y otros, 2014)**:**

Contienen código máquina, y no código fuente, por lo que cada tipo de arquitectura (X86((i386-i686), AMD64[2], ARM[3]), necesita su propio paquete. Estos pueden ser:

[2] Arquitectura de una computadora que posee una versión de 64 bits

- ✓ RPM: Estos paquetes son utilizados por distribuciones como *Red Hat, Suse, Mandrake*, *Conectiva*, Caldera.
- ✓ DEB: Estos paquetes son utilizados por distribuciones como *Debian*, y las basadas en ella, como *Ubuntu*. Las utilidades para manejar este tipo de paquetes son apt y dpkg.
- ✓ TGZ: Son utilizados por la distribución *Linux Slackware*.

Paquetes de código fuente

Contienen el código fuente del programa, estos vienen con los archivos necesarios para compilar e instalar el programa manualmente. Suelen presentarse en formato tar.gz o tar.bz2 (o sea compactado con tar y comprimido con gzip o bzip). Normalmente cada aplicación tiene la información en el fichero *README* o *INSTALL* de cómo instalarlo (Miranda Domínguez, y otros, 2014).

1.2 *Logs*

Necesidad de analizar los *logs*

Los registros nos proporcionan la información necesaria sobre cómo se está comportando nuestro sistema. Sin embargo, el contenido y el formato de los registros varían entre los diferentes servicios o decir, entre los diferentes componentes del mismo sistema. Por ejemplo, un escáner puede registrar mensajes de error relacionados con la comunicación con otros dispositivos; Por otro lado, un servidor *web* registra información sobre todas las solicitudes entrantes, respuestas salientes, tiempo tomado para una respuesta, y así sucesivamente.

Del mismo modo, los registros de aplicación de un sitio *web* de comercio electrónico registrarán registros empresariales específicos. Como los registros varían según su contenido, también lo harán sus usos. Por ejemplo, los registros de un escáner pueden utilizarse para la resolución de problemas o para una comprobación o informe de estado simple mientras el registro del servidor *web* se utiliza para analizar patrones de tráfico en varios productos. El análisis de registros de un sitio de comercio electrónico puede ayudar a consultar si los paquetes de una ubicación específica se devuelven repetidamente y las razones probables de la misma. Los siguientes son algunos casos de uso comunes en los que el análisis de *log* es útil: (Chhajed, 2015)

---

[3] Arquitectura RISC (Ordenador con Conjunto Reducido de Instrucciones) de 32 bits

• Problema de depuración

• Análisis de rendimiento

• Análisis de seguridad

• Análisis predictivo

• Internet de las cosas (*IoT*) y registro

Análisis de rendimiento

El análisis de registros ayuda a optimizar o depurar el rendimiento del sistema y aportar datos esenciales sobre cuellos de botella en el sistema. Comprender el rendimiento de un sistema es a menudo sobre la comprensión del uso de recursos en el sistema. Los registros pueden ayudar a analizar el uso de recursos individuales en el sistema, el comportamiento de varios subprocesos en la aplicación, las posibles condiciones de interbloqueo. Los registros también llevan consigo información de marca de tiempo, que es esencial para analizar cómo se está comportando el sistema con el tiempo. Por ejemplo, un registro del servidor *web* puede ayudar a saber cómo se desempeñan los servicios individuales en función de los tiempos de respuesta y los códigos de respuesta *HTTP*. (Chhajed, 2015)

Análisis predictivo

El análisis predictivo es una de las tendencias más relevante de los últimos tiempos. Los registros y los datos de eventos se pueden utilizar para un análisis predictivo muy preciso. Los modelos de análisis predictivo ayudan a identificar clientes potenciales, planificación de recursos, administración y optimización de inventario, eficiencia de la carga de trabajo y programación eficiente de recursos. (Chhajed, 2015)

1.2.1 Estructura del archivo de formato *log*

Los archivos de formato *log* registran todos los accesos a su alojamiento, guardando información acerca de la dirección *IP* desde la que se ha realizado la conexión, fecha, hora, archivos y/o página a la que se accede. Estos ficheros, que se encuentran en la ruta ***/var/log/*** (en la mayoría de las distribuciones), tienen la función de almacenar los registros del sistema, información de gran utilidad para producir estadísticas de visitas (Bahit, 2012).

Para poder leer y entender la información que se desea mostrar es necesario estudiar la estructura de los archivos de formato *log,* para ello se analizó el del servidor *web* Apache. Existen dos tipos de formato: *common y combined;* pero en este caso se analizará *combined* por poseer una estructura más abarcadora en sentido general.

Estructura de *combined*: "%h %l %u %t \"%r\" %>s %b \"%{Referer}\" \"%{User-agent}\""

- %h: host que accede (dirección *IP*).
- %l: protocolo de identificación del usuario RFC 1413, aparece también como un guión (-).
- %u: nombre del usuario (si el usuario no está registrado en el sistema la salida sería un guión).
- %t: marca de tiempo, fecha completa incluyendo la hora (20/Oct/2016:10:23:58 -0300.)
- %r: solicitud realizada por el usuario, se incluye el método GET, POST, PUT, otros, y el archivo al cual se accede. Ejemplo: (“GET /novarepo.php HTTP/1.1").
- %s: código de respuesta de estado. Ejemplo: 200 (*OK*), 404 (*Not Found*).
- %b: cantidad de *bytes* entregados al usuario.
- %{Referer} es el archivo, sitio o página que contiene el vínculo hacia el recurso solicitado. Ejemplo: http://example.org/goTo.php? p=http://example.com/index.php.
- %{User-agent} representa el user-agent del usuario. Por ejemplo.: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv: 11.0) Gecko/20100101 Firefox/11.0.

Ejemplo de la estructura completa:

10.8.112.242 - - [26/Feb/2017:14:21:38 -0500] "*GET* nova/2015/pool/extendido/t/tinyxml/libtinyxml2.6.2_2.6.2-2_amd64.deb HTTP/1.1" 200 35259 "-" "Debian *APT-HTTP*/1.3 (1.0.1ubuntu2)"

1.3 Procesamiento estadístico.

El procesamiento estadístico no refiere a una técnica, a un algoritmo o a un procedimiento particular. Es una filosofía de la optimización referida a mejoras de proceso continuas, usando una colección de las herramientas (estadísticas) para datos y análisis del proceso. Es un componente dominante de las iniciativas totales de la calidad (Estévez, 2009).

Las herramientas comúnmente usadas en el procesamiento estadístico son:

1. Organigramas.
2. Gráficas de funcionamiento.
3. Gráfica y análisis de pareto.
4. Diagramas de causa-efecto.
5. Histogramas de la frecuencia.
6. Gráficas de Control.
7. Estudios de la capacidad de proceso.
8. Planes de muestreo de aceptación.
9. Diagramas de dispersión.

Procesamiento estadístico descriptivo

Consiste en recolectar los datos, organizarlos, presentarlos, analizarlos e interpretar los resultados con el objetivo que los datos se comprendan más fácilmente y que sea más sencillo referirse a ellos (Romero, 2002).

De acuerdo con la definición anteriormente expuesta, el autor de la presente investigación científica considera que los procesamientos estadísticos descriptivos son técnicas que describen, analizan y representan a un grupo de datos utilizando métodos gráficos y numéricos que resumen y prestan la información contenida en ellos.

Técnicas gráficas

Es la representación en el plano de la información estadística, con el fin de obtener una impresión visual global del material representado, que facilite su rápida comprensión. Los gráficos pueden servir no solo como sustituto a las tablas, sino que también constituye por sí mismo una poderosa herramienta para el análisis de los datos, siendo en ocasiones el medio más efectivo no solo para describir y resumir la

información, sino también para analizarla (Sneiderman, y otros, 2015).

De acuerdo con las definiciones anteriormente expuestas, el autor de la presente investigación científica considera que las técnicas gráficas son maneras de representar los datos de forma que brinden información a primera vista.

1.4 Herramientas para el análisis de archivos de formato *log*

Las herramientas para el análisis de *logs* ayudan a monitorear la integridad de los archivos, llevar a cabo análisis forenses informáticos, monitorear usuarios privilegiados y cumplir con diferentes entidades regulatorias al analizar inteligentemente sus *logs* y generar instantáneamente una variedad de informes como de actividad de usuarios y tendencias. A continuación, se analizan una serie de herramientas con el objetivo de seleccionar cual es la más adecuada para la creación del sistema informático de monitoreo en tiempo real al uso del repositorio de Nova.

1.4.1 *Webalizer*

Es un programa que genera estadísticas del sitio *web*, el cual permite el análisis de los datos obtenidos del *log* del servidor web. Muestra todo tipo de estadísticas de acceso al sitio, desde el navegador utilizado hasta el origen de la visita (buscadores, acceso directo, enlaces desde otros sitios). También informa del país de donde se accede, el tiempo medio de visita y las palabras claves utilizadas para llegar al sitio a través de un buscador *web*. Es una herramienta útil ya que registra el acceso real al servidor sin importar si su origen es un navegador, un robot de rastreo (buscadores) o un enlace directo al documento (pdf u otros). *Webalizer* da una visión macro de las páginas y documentos consultados. La información la presenta en categorías como Acceso, Archivos, Páginas, Clientes, Visitas y *Kbytes*. Unos de los aspectos negativos que presenta la herramienta son que no es una herramienta personalizable y la interfaz de usuario posee baja usabilidad (Degiorgi, y otros, 2011).

1.4.2 *Google Analytics*

Es un servicio gratuito de estadísticas de sitios *web* que ofrece la empresa *Google*. Agrupa la información y permite trabajar con variables personalizadas que ayudan a definir segmentos de seguimiento múltiples, e incluso simultáneos, según los datos de resultados de búsqueda, sesiones o visitas. Hace énfasis principalmente en el comportamiento del usuario frente a la página. Es una herramienta muy completa que

si bien está orientada a *webs* comerciales es mucha la información que puede aportarnos para la toma de decisiones (Degiorgi, y otros, 2011).

1.4.3 *Piwik*

Aplicación de código abierto que requiere ser instalado en el servidor y registra el tráfico en tiempo real. En muchos aspectos es similar a *Google Analytics*, pero la principal ventaja es que uno es el propietario de los datos registrados de los visitantes, los cuáles se almacenan en una base de datos. Además, permite ejecutar el sistema dentro del mismo dominio, lo cual hace que ciertas restricciones de seguridad sean subsanadas (Degiorgi, y otros, 2011).

1.4.4 *ELK Stack*

Es principalmente el conjunto de tres herramientas (*Elasticsearch*, *Logstash* y *Kibana*) que se utilizan de forma conjunta para gestionar, almacenar y representar datos en tiempo real.

Elasticsearch

Motor de búsqueda y análisis de código abierto altamente escalable. Permite indexar un gran volumen de datos y posteriormente hacer consultas sobre ellos soportando entre otras muchas cosas búsquedas aproximadas, facetas y resaltado. Un uso puede ser hacer consultas de texto completo, al estar los datos indexados los resultados se obtienen de forma muy rápida. *Elasticsearch* es capaz de comunicarse mediante su interfaz *web* con otros sistemas y además guarda la información en documentos formato *JSON* (Elasticsearch, 2015). Esta herramienta cubre las necesidades de procesar los datos, de almacenarlos y de transmitirlos a otros componentes.

Logstash

Conducto de datos que es empleado para la recolección, el enriquecimiento y el transporte de datos de manera abierta y fiable. Con conectores a una infraestructura común para una fácil integración, *Logstash* está diseñado para procesar eficientemente una creciente lista de datos, eventos y fuentes de datos no estructurados para su distribución en una variedad de productos, incluyendo *Elasticsearch*. Esta herramienta se trata de una tubería, esto quiere decir que se divide en tres componentes en su configuración: **entrada** o ***input*** (por donde entra la información), **filtro** o ***filter*** (como se trata la información) y **salida** u ***output*** (por donde sale la información) (Logstash, 2015). Esta herramienta realiza la función de obtención, análisis, modificación y reenvío de información a *Elasticsearch*.

Kibana

Herramienta capaz de manejar la información que hay en *Elasticsearch*. Muestra distintos diagramas de visualización, en él se puede configurar uno o varios *dashboard* con la información que requieran los usuarios. Además, permite exportar los resultados obtenidos y posee la capacidad de realizar búsquedas y filtros sobre los datos guardados en *Elasticsearch* (Kibana, 2015).

1.4.5 *Grafana, Graphite e InfluxDB*

Grafana

Herramienta de código abierto que trabaja comúnmente con *Graphite, InfluxDB* y *OpenTSDB*. Es empleada para la visualización de información analizada y procesada de un ambiente determinado. *Grafana* estructura las diferentes vistas en los llamados *dashboards*. Cada *dashboard* está formado por diferentes paneles (gráficos, tablas, *HTML*) en los que a través de un *query builder* es capaz de recoger los datos de diferentes *backends.* Dichos datos se obtienen mediante una llamada *HTTP* a los diferentes *datasources* soportados por *Grafana*: uno de ellos es *Graphite*. (Grafana Labs, 2017)

Graphite

Herramienta de monitoreo para empresas que funciona igualmente bien con *hardware* o infraestructura de *Cloud*. *Graphite* se utiliza para realizar un seguimiento del rendimiento en sitios *web*, aplicaciones, servicios empresariales y servidores en red. Mediante la herramienta se pueden almacenar, recuperar, compartir y visualizar datos de series de tiempo. (Graphite, 2017)

InfluxDB

Soporta altas cargas de escritura, almacenamiento de grandes conjuntos de datos y conserva espacio a través del muestreo, expiración y eliminación automática de datos no deseados, así como copia de seguridad y restauración. *InfluxDB* también facilita el análisis de datos proporcionando un lenguaje de consulta similar al SQL de fácil uso.

Puede graficar y visualizar sus datos con el proyecto de código abierto *Chronograph*, y realizar la exploración *ad hoc* de sus datos. Incluye soporte para plantillas y una biblioteca de cuadros de mandos inteligentes y preconfigurados para conjuntos de datos comunes. Además, *InfluxDB* soporta otras herramientas como *Grafana*. (DB-Engines, 2017)

1.5 Comparación entre los sistemas de análisis de archivos de formato *log*

En la siguiente tabla se representa una comparación de las principales características de los sistemas de análisis descritos anteriormente. Para ello se relacionan algunos aspectos tomados de la Norma Cubana ISO/IEC 9126 (Oficina Nacional De Normalización, 2005).

Tabla 1: Comparación entre las diferentes herramientas de análisis de *logs*.

Herramientas	Webalizer	Google Analytics	Piwik	ELK Stack
Sistema Operativo	Linux	Multiplataforma	Multiplataforma	Multiplataforma
Licencia	GNU/GPL	GNU/GPL	GNU/GPL	GNU/GPL
Idioma	Multilingüe	Multilingüe	Multilingüe	Inglés
Estado	Completo	Completo	Completo	Completo
Comunidad activa	Si	Si	Si	Si
Proyecto activo	Si	Si	Si	Si
Tipo de aplicación	Web	Web	Web	Web
Lenguajes de programación	C	JavaScript	PHP	Java
Servidor web	Apache	Apache	Multiservidor	Apache

Tabla 2: Comparación entre las diferentes herramientas de análisis de *logs*.

Herramienta	Grafana	Graphite	InfluxDB
Sistema Operativo	Multiplataforma	Linux, Unix	Linux, OS X

Licencia	GNU/GPL	Open Source	Open Source
Idioma	Inglés	Inglés	Inglés
Estado	Completo	Completo	Completo
Comunidad activa	Si	Si	Si
Proyecto activo	Si	Si	Si
Tipo de aplicación	Web	Web	Web
Lenguajes de programación	Java	Python	PHP, Java, JavaScript, JavaScript (Node.js)
Servidor web	Apache	-	-

1.6 Resultado de la comparación entre los sistemas de análisis de archivos de formato *log*

Luego del estudio realizado a las herramientas de análisis de archivos de formato *log* se evidencian notables semejanzas, pero aun así existen características en algunas de ellas que imposibilitan su uso para el desarrollo del proyecto.

- *Webalizer* produce cifras de visitas irreales, no dispone de resultados semanales, no es una herramienta personalizable y la interfaz de usuario posee baja usabilidad. Por otra parte, no analiza los paquetes de formato .deb y tampoco utiliza gestores de base de datos, ya que almacena los datos en texto plano.

- *Piwik* es una herramienta superior en cuanto a todos los aspectos antes mencionados, es de las más utilizadas en el mundo y además utiliza *PHP* como lenguaje de programación, pero para usarla es necesario colocar un código *JavaScript* que proporciona *Piwik* en cada página a la que se le quiere dar seguimiento, lo mismo sucede con *Google Analystics*, por tanto, sería engorroso ir colocando en cada directorio del repositorio una página *HTML* que pudiera hacer ese seguimiento.

- El conjunto de herramientas *Grafana, Graphite* e *InfluxDB* se enfoca en la presentación de gráficos de series de tiempo, basados en métricas específicas tales como la *CPU*[4] y la utilización de E/S[5]. *Kibana*, por otro lado, se ejecuta en la parte superior de *Elasticsearch* y puede crear un completo tablero de análisis de *logs*. Por ejemplo, *Grafana* no permite la búsqueda y exploración de datos.

Según *Google Trends,* se presentan a continuación estadísticas sobre el empleo de *ELK Stack* por las comunidades de código abierto.

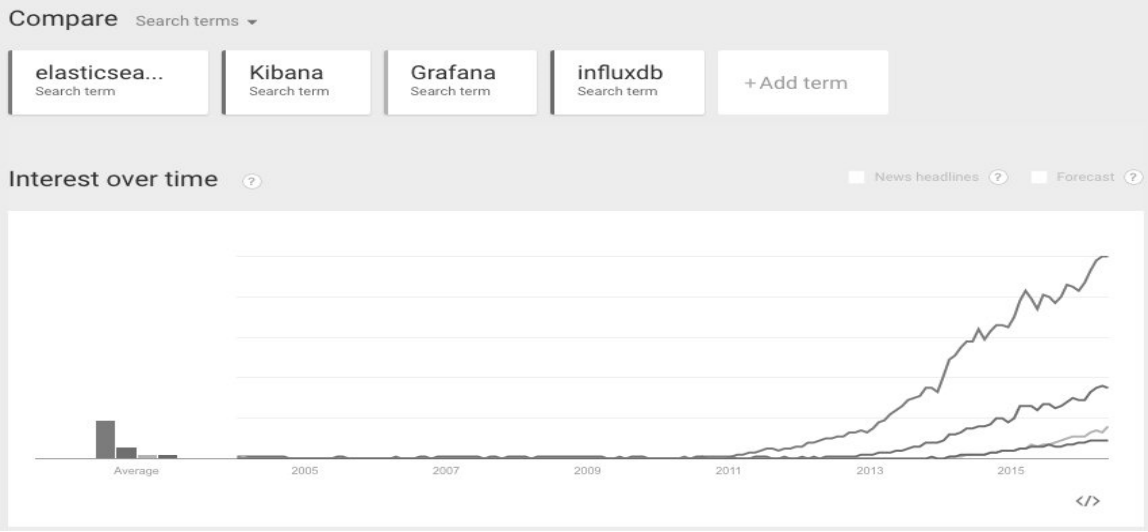


Ilustración 1: Comparativa entre las tecnologías *ELK Stack* y *Grafana*

Para el desarrollo del proyecto se decidió que la tecnología a emplear es *ELK Stack* porque facilita el procesamiento en tiempo real de diferentes tipos de registros sin importar su estructura, provee una búsqueda escalable, soporta facetado y percolación, que puede ser útil para notificar si nuevos documentos coinciden con consultas registradas anteriormente.

1.7 Metodología y herramientas para la elaboración del sistema

Metodología de desarrollo

Para el desarrollo de un *software* es imprescindible el empleo de una metodología, la cual guiará todo el proceso de concepción del producto. Aunque en la implementación del sistema no se define la

[4] Expresión inglesa *Central Processing Unit*, en español Unidad Central de Proceso.

[5] Dispositivos de entrada y salida.

construcción de un *software* en su ciclo completo de desarrollo, se determinó el uso de la metodología AUP-UCI que posee 3 fases:

- **Inicio:** Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
- **Ejecución:** En esta fase se ejecutan las actividades requeridas para desarrollar el *software*, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.
- **Cierre:** En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

De los escenarios que presenta la metodología se escogió el número 4, se realiza todo el modelado de la solución para un mejor entendimiento y se elaboran los requisitos, utilizando así las historias de usuario para luego poder implementarlos, probarlos y validarlos.

Servidor web

Para la implementación del sistema se utilizará el servidor *web* Apache. Es una aplicación libre y de código abierto, ejecutable en varios sistemas operativos: *Unix, FreeBSD, Linux, Solaris, Novell NetWare, OS X, Microsoft Windows*. Permite múltiples lenguajes de script: PHP, Perl, Tcl y Python. Posee una configuración sencilla basada en directivas que se editan en ficheros: httpd.conf, access.conf (en *Unix*), .htpasswd (Mestras, 2013).

Herramientas

Herramientas CASE Visual Paradigm

Las herramientas CASE (***C****omputer* ***A****ided* ***S****oftware* ***E****ngineering*, Ingeniería de *Software* Asistida por Ordenador) permiten construir diagramas como el diagrama de paquetes y el diagrama de despliegue. Visual Paradigm para BPMN posibilita un ahorro considerable de tiempo y una calidad óptima en el modelado de procesos del negocio. Se considera muy completa y fácil de usar, con soporte multiplataforma (Díaz, 2009).

Sublime Text

Es un editor de texto y editor de código fuente, está escrito en C++ y Python para los *plugins*. Soporta de forma nativa 43 lenguajes de programación y el remarcado de sintaxis es completamente configurable a través de archivos de configuración del usuario. Posee un intérprete de *Python* diseñado solo para el programa con el cual se puede realizar infinidad de tareas, si se escribe en un lenguaje de programación o marcado, resalta las expresiones propias de la sintaxis de ese lenguaje para facilitar su lectura. (Sublime Text, 2017)

RegExr

Herramienta en línea para aprender, construir y probar expresiones regulares. Posibilita resultados actualizados en tiempo real, brinda además información detallada de cada expresión regular. (RegExr, 2017)

Valoraciones generales del capítulo

En este capítulo, con el objetivo de darle solución al problema planteado se realizó un estudio de diferentes tipos de herramientas de análisis de *logs* existentes. El mismo permitió definir a la tecnología *ELK Stack* como la más indicada para la implementación de un sistema informático de monitoreo en tiempo real que permita conocer las estadísticas acerca de la instalación de las aplicaciones del repositorio de Nova.

Capítulo 2. Descripción de la solución propuesta

Introducción

En el presente capítulo se analiza y diseña la propuesta de solución del sistema a implementar guiada por la metodología de desarrollo de *software* AUP-UCI. Se describen las características que debe cumplir el sistema, definiéndolas a través de los requisitos funcionales y no funcionales.

2.1 Solución propuesta

Se propone la implementación de un sistema informático de monitoreo en tiempo real que permita conocer las estadísticas acerca de la instalación de las aplicaciones del repositorio de Nova. Para ello serán empleadas las herramientas *Elasticsearch, Logstash* y *Kibana*.

En *Logstash*, los *plugins* de **entrada (*input*)** habilitan el soporte para diversas fuentes generadoras de *logs*. Los *plugins* de **filtro (*filter*)** permiten el procesamiento, incluidos filtrado y normalización, de los *logs* recolectados en las fuentes. Los *plugins* de **salida (*output*)** posibilitan el envío de los *logs* recolectados y procesados a su destino final.

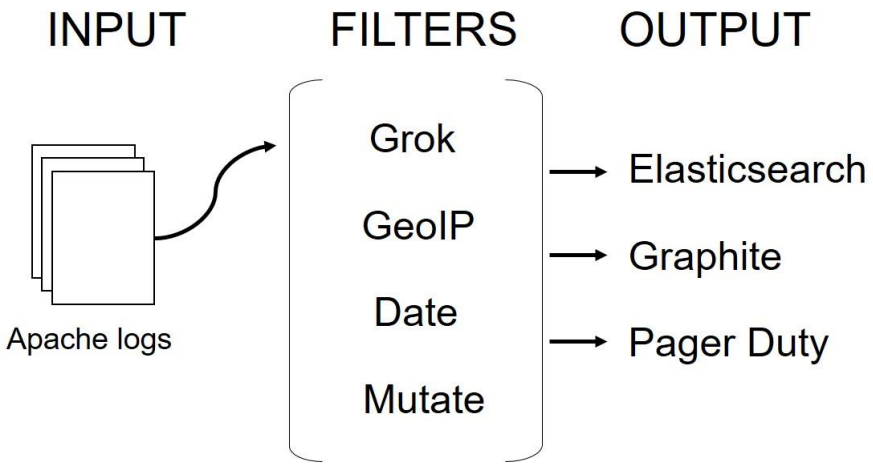


Ilustración 2: Arquitectura de *Logstash*. Fuente: elaboración propia.

En el sistema a desarrollar *Elasticsearch* cubre las necesidades de procesar los datos, de almacenarlos y de transmitirlos hacia otros componentes. *Logstash* realiza la función de obtención, análisis, modificación y reenvío de la información a *Elasticsearch*. Finalmente *Kibana* permite la visualización y monitorización de las unidades de información procesadas. Kibana actúa como *frontend*, mientras *Elasticsearch* y *Logstash* actúan como *backend*.

El sistema obtendrá los datos del archivo de formato *log* generado por el acceso al servidor donde se encuentra alojado el repositorio de Nova. Se configurará *Logstash* de manera que se analice y se procese la información contenida en el archivo de formato *log,* extrayendo así los valores requeridos que posteriormente se mostrarán con el visualizador *Kibana* para realizar el proceso estadístico mediante tablas y gráficas.

A continuación, se muestra en la Ilustración 3 el modelo de relación entre los paquetes del análisis del sistema. En dicho modelo se detallan los principales componentes y las relaciones que se establecen entre ellos.

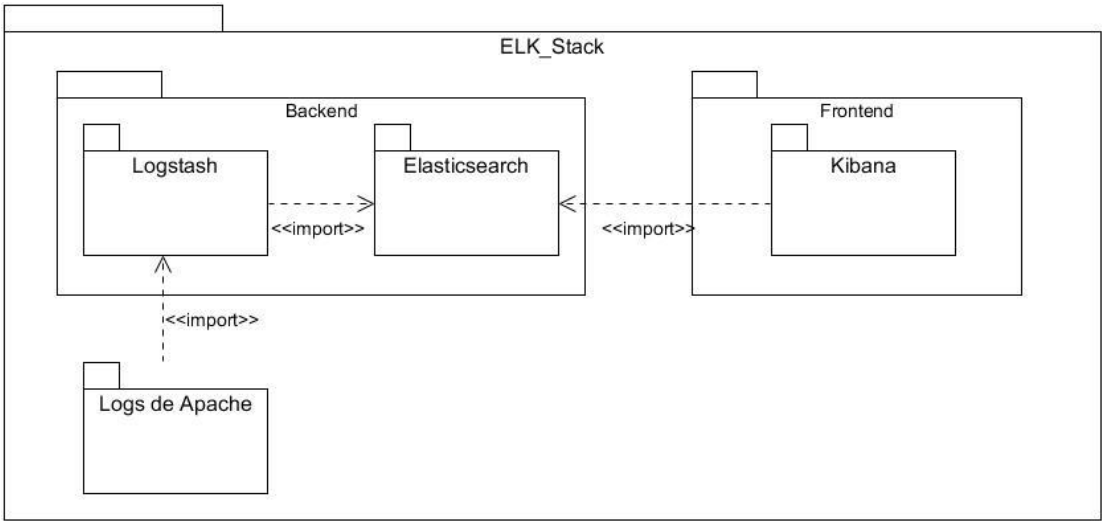


Ilustración 3: Diagrama de paquetes de análisis del diseño de ELK Stack. Fuente: elaboración propia.

Los resultados que se generarán son los siguientes:

La presentación visual de información en *Kibana* se basa en gráficos (*visualization*) agrupados en paneles que componen un cuadro de mandos o *dashboard*. Una vez iniciada la herramienta, en las pantallas:

Discover: Se mostrarán los resultados obtenidos, mostrando por cada acceso al repositorio lo siguiente: IP de acceso, fecha, hora y el paquete descargado.

Visualize: Es donde se pueden crear y modificar visualizaciones personalizadas (gráficos y tablas), a raíz de las búsquedas que se realicen en *Kibana* sobre los elementos indexados en *Elasticsearch* y permite la elección del tipo de gráfica que se requiera, además de la disposición de los datos.

Dashboard: Pantalla donde se pueden crear, modificar y ver cuadros de mando personalizados, pudiendo guardar dichos *dashboards* y cargarlos según las necesidades.

Settings: Pantalla que permite cambiar la configuración por defecto o patrones de índice (En este caso solo se empleará el índice ***logstash**).

2.2 Especificación de requisitos de *software*

En la ingeniería del *software*, los requisitos se utilizan como datos de entrada en la etapa de diseño del producto y establecen qué debe hacer el sistema, pero no cómo hacerlo. Son una condición o capacidad que un usuario necesita para poder resolver un problema o lograr un objetivo. De manera general estos requisitos son lo que el sistema debe hacer o una cualidad que el sistema debe poseer (Somerville, 2005).

2.2.1 Requisitos funcionales (RF)

Un requisito funcional define una función del sistema de *software* o sus componentes.

Tabla 3: Especificación de los requisitos funcionales.

Código	Descripción (Requisitos funcionales)	Prioridad
RF1	Añadir *url* de la ubicación del archivo *log* de Apache	Alta
RF2	Definir la búsqueda a descargas de archivos con extensión .deb	Alta
RF3	Mostrar cantidad de descargas por fecha	Alta
RF4	Mostrar las direcciones *IP* que más paquetes descargan del repositorio	Alta
RF5	Mostrar cantidad de descargas por paquetes	Alta
RF6	Mostrar cantidad de paquetes descargados por dirección *IP*	Alta
RF7	Mostrar las 10 primeras direcciones *IP* que más paquetes descargan del repositorio	Alta
RF8	Mostrar gráficas de tendencias por semana	Baja
RF9	Mostrar gráficas de tendencias por meses	Baja
RF10	Mostrar gráficas de tendencias por años	Baja

2.2.2 Requisitos no funcionales (RNF)

Los requisitos no funcionales o requerimientos no funcionales no son más que limitaciones sobre servicios o funciones que ofrece el sistema. Incluyen restricciones tanto de temporización y del proceso de desarrollo, como impuestas por los estándares. Los requerimientos no funcionales se suelen aplicar al sistema como un todo, más que a características o a servicios individuales del sistema (Somerville, 2005). Son los encargados de darle el acabado al producto para que esté listo para usarse. Se identificaron un total de 3 requisitos no funcionales, agrupados en 3 categorías que a continuación se presentan.

Requerimientos de *software*

RNF1: Instalación de la máquina virtual de Java (JVM) en su versión 7 como mínima requerida.

Requerimientos de espacio

RNF2: La estación de trabajo debe contar con un mínimo de 500 GB de espacio disponible en el disco duro.

Requerimientos de rendimiento

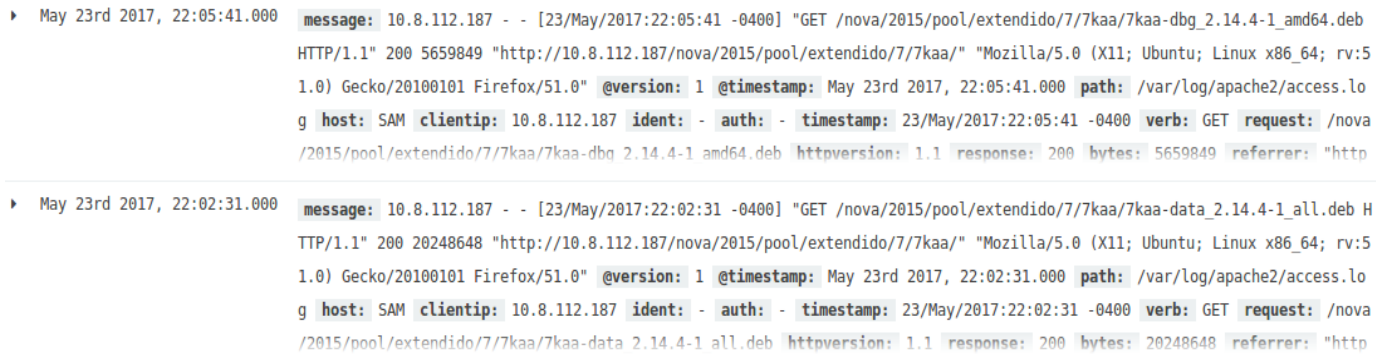
RNF3: La estación de trabajo donde se despliega el sistema debe poseer un procesador Dual Core o superior, con un procesamiento a 1.48 GHz o superior y 4GB de RAM o superior.

2.3 Historias de Usuario (HU)

Las HU se utilizan para registrar los requerimientos de los clientes según el negocio y son utilizadas para poder realizar la estimación de cada una de las iteraciones durante la fase de planificación.

Tabla 4: Definir la búsqueda a descargas de archivos con extensión .deb.

Historia de Usuario	
Número: HU_2	**Nombre:** Definir la búsqueda a descargas de archivos con extensión .deb
Prioridad: Alta	**Iteración asignada:** 2
Programador: Raúl Sánchez Arañó	**Tiempo estimado:** 1 día

Riesgo en desarrollo: - Fallo de conexión - Fallo del servidor	**Tiempo:** 1 día
Descripción: Se configurará *Logstash* para que analice las líneas de *logs* buscando descargas de archivos con extensión .deb, o sea, las aplicaciones que se descargan por los usuarios.	
Prototipo:  ▸ May 23rd 2017, 22:05:41.000 **message:** 10.8.112.187 - - [23/May/2017:22:05:41 -0400] "GET /nova/2015/pool/extendido/7/7kaa/7kaa-dbg_2.14.4-1_amd64.deb HTTP/1.1" 200 5659849 "http://10.8.112.187/nova/2015/pool/extendido/7/7kaa/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:51.0) Gecko/20100101 Firefox/51.0" **@version:** 1 **@timestamp:** May 23rd 2017, 22:05:41.000 **path:** /var/log/apache2/access.log **host:** SAM **clientip:** 10.8.112.187 **ident:** - **auth:** - **timestamp:** 23/May/2017:22:05:41 -0400 **verb:** GET **request:** /nova/2015/pool/extendido/7/7kaa/7kaa-dbg_2.14.4-1_amd64.deb **httpversion:** 1.1 **response:** 200 **bytes:** 5659849 **referrer:** "http ▸ May 23rd 2017, 22:02:31.000 **message:** 10.8.112.187 - - [23/May/2017:22:02:31 -0400] "GET /nova/2015/pool/extendido/7/7kaa/7kaa-data_2.14.4-1_all.deb HTTP/1.1" 200 20248648 "http://10.8.112.187/nova/2015/pool/extendido/7/7kaa/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:51.0) Gecko/20100101 Firefox/51.0" **@version:** 1 **@timestamp:** May 23rd 2017, 22:02:31.000 **path:** /var/log/apache2/access.log **host:** SAM **clientip:** 10.8.112.187 **ident:** - **auth:** - **timestamp:** 23/May/2017:22:02:31 -0400 **verb:** GET **request:** /nova/2015/pool/extendido/7/7kaa/7kaa-data_2.14.4-1_all.deb **httpversion:** 1.1 **response:** 200 **bytes:** 20248648 **referrer:** "http	

2.3 Diagrama de despliegue

El diagrama de despliegue es un tipo de diagrama del Lenguaje Unificado de Modelado que se utiliza para modelar la disposición física de los artefactos software en nodos (usualmente plataforma de *hardware*) (OMG UML, 2007). Para un mayor entendimiento de cómo se desplegará el sistema, ver Ilustración 4.

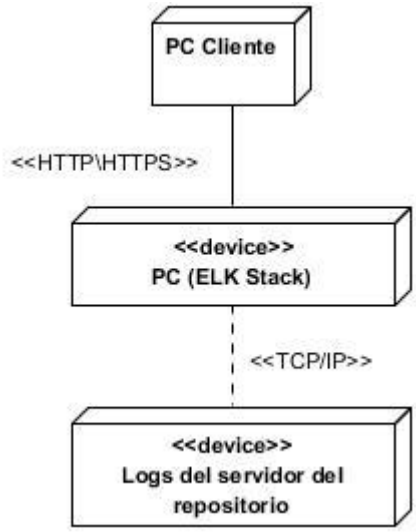


Ilustración 4: Diagrama de despliegue del sistema. Fuente: elaboración propia.

Valoraciones generales del capítulo

En este capítulo se mostró una visión de cómo quedaría el sistema de análisis de *logs*, utilizando para ello la metodología de desarrollo AUP-UCI. La solución propuesta muestra cómo se realiza el proceso de análisis de archivos de formato *log*, *Logstash* se encarga de procesar el fichero, envía los datos hacia Elasticsearch, que posteriormente serán indexados y luego *Kibana* visualizará en gráficos y tablas la información requerida. Se precisaron un total de 13 requisitos, de los cuales 10 son RF y 3 RNF.

Capítulo 3. Implementación y prueba

Introducción

La etapa de implementación constituye una parte crucial durante el proceso de desarrollo de un *software*, es en este momento donde se define y organiza el código de la propuesta de solución. Durante esta etapa se materializan en forma de código las descripciones y arquitectura propuestos en la etapa de análisis y diseño; conformando así el producto final requerido por el cliente.

Todo *software* debe ser puesto a prueba, garantizando así que cumpla con todos los estándares requeridos y con todas aquellas condiciones y funcionalidades que el cliente final necesita. A esta etapa se le conoce como validación del sistema y en ella, pueden realizarse diferentes tipos de pruebas en función de los objetivos de las mismas.

3.1 Estándares de codificación utilizados

Una buena práctica de programación que potencia la legibilidad del código es la utilización de estándares de codificación. Un estándar de codificación comprende todos los aspectos de la generación de código y la legibilidad del mismo.

A continuación, se mencionan las principales prácticas utilizadas para la configuración de *Logstash.*

Identación: Emplear 4 espacios por cada nivel de identación.

```
input {

    file {

    path => "/var/log/apache2/access.log"
    start_position => "beginning"
    }

}
```

Ilustración 5: Configuración para los datos de entrada de *Logstash*. Fuente: elaboración propia.

Tabuladores y espacios: No mezclar las tabulaciones con los espacios

```
filter {

    if ".deb" in [request]{

        grok {
        # Aqui se comprueba si la línea de registro contiene una fecha

        match => { "message" => "%{COMBINEDAPACHELOG}" }
        add_field => { "paqueteDescargado" => "" }
        }

        date {
        match => [ "timestamp" , "dd/MMM/yyyy:HH:mm:ss Z" ]
        target => "@timestamp"
        }
    }

    mutate {

      gsub => ['Paquete_descargado', "\t", " "]

     }


}
```

Ilustración 6: Configuración del *plugin filter* para el procesado de los *logs*. Fuente: elaboración propia.

Declaraciones: Se debe declarar cada *plugin* y filtro en una línea diferente, de esta forma se puede comentar cada uno por separado.

Ilustración 7: Declaración de los *plugins* y filtros en la configuración de *Logstah*. Fuente: elaboración propia.

Tamaño máximo de líneas: Se limitan todas las líneas a un máximo de 50 caracteres. Esto puede ser realizado mediante el uso de paréntesis de forma implícita o mediante el uso de la barra invertida.

Comentaros: Los comentarios deben explicar de manera concisa el código, aportándole mayor claridad al lector.

Líneas en blanco: Las definiciones de *plugins* en la configuración son separados por una línea en blanco.

```
input {

}

filter {

}

output {

}
```

Ilustración 8: *Plugins* de entrada, procesado y salida de los datos. Fuente: elaboración propia.

3.2 Iniciando y configurando la tecnología *ELK Stack*

3.2.1 Inicialización y configuración de *Elasticsearch*

La herramienta *Elasticsearch* no requiere de configuración adicional para el desarrollo de la solución propuesta. Sin embargo, se pueden realizar modificaciones en la configuración si el cliente lo desea, para ello existe un fichero llamado **"elasticsearch.yml"**:

- ***host*:** localhost
- ***port*:** 9200 (es el que tiene por defecto)
- ***cluster.name*:** ELK_Nova (nombre del clúster)
- ***node.name*:** Nova_Repositorio (nombre del nodo)

3.2.2 Inicialización y configuración de *Logstash*

La instancia de *Logstash* debe ser lanzada junto con un fichero de configuración que define el comportamiento que va a tener y cómo va a tratar cada línea del fichero *log*.

Entrada de información

Los *logs* entran a *Logstash* por filas, por lo que no se procesa la siguiente fila hasta que se haya terminado de procesar la anterior, a menos que la herramienta se ejecute en varios hilos.

La entrada de información está definida por el parámetro "***input* {}**". Dado que la fuente de información especificada es un fichero *log*, la entrada de datos está orientada a ficheros, por lo que se necesita el

plugin "***file* {}**".

El *plugin* ***file*** se ha de indicar de la siguiente manera:

- ***path:*** ruta completa al fichero *log* de Apache.
- ***start_position:*** desde donde se quiere analizar el fichero, existiendo dos opciones:
    - ***beginning:*** desde el inicio del fichero.
    - ***end:*** desde el final del fichero.

Se ha elegido como posición inicial desde el principio del fichero para poder trabajar con todos los datos del fichero una vez que *Logstash* se haya iniciado.

Procesamiento de los *logs*

El procesamiento se realiza con el *plugin* "***filter* {}**", para poder extraer y diferenciar los elementos requeridos de los *logs*, además de hacer modificaciones sobre ellos. Para realizar el análisis del archivo se emplea el *plugin* "***grok* {}**", en el cual se detalla el procesamiento de los datos mediante una sentencia en *regrex* (expresión regular). En este caso se emplea el patrón "**%{*COMBINEDAPACHELOG*}**" que es para registros apaches que siguen el formato de registro combinado (***combined* apache *log***) y es el que viene por defecto en la distribución GNU/Linux Nova, se usó a través de la opción "***match***" la cual permite asignar uno o varios valores a un campo en específico, en este caso al campo "***message***" se le asignó el evento completo.

Además se empleó el filtro "***date***" el cual permite por medio de la opción "***match***" asignarle al campo "***timestamp***" la fecha en que se generó el evento siguiendo el formato **"dd/MMM/yyyy:HH:mm:ss Z**". El campo "***@timestamp***" es el campo que representa la marca de tiempo por defecto cuando el evento fue capturado por *Logstash*, y el cual será usado por *Kibana* para mostrar los eventos según la fecha, lo que interesa aquí es que este contenga la marca de tiempo del evento o sea la que contiene el campo "***timestamp***", esto se logra con la opción "***target***".

Salida de la información

La salida está definida por "***output* {}**", donde se utilizó el *plugin* **"elasticsearch"** para la salida de los datos procesados al servidor, cuya localización se indica en el atributo ***"hosts"***. En este caso *Elasticsearch* se indica en *localhost*.

3.2.3 Inicialización y configuración de *Kibana*

La herramienta *Kibana* debe de ejecutarse con permisos de administración, no requiere de una configuración adicional. Sin embargo, la configuración se realiza mediante el fichero **"kibana.yml"**:

- ***host:*** 0.0.0.0 (para cualquier conexión de entrada)
- ***port:*** 5601 (es el que tiene por defecto)
- ***host* de *Elasticsearch*:** http://localhost:9200 (*host* de conexión con el servicio)

3.3 Pruebas de *software*

Las pruebas de *software* son un conjunto de herramientas, técnicas y métodos que evalúan la excelencia y el desempeño de un *software*, involucra las operaciones del sistema bajo condiciones controladas y evaluando los resultados. Las técnicas para encontrar problemas en un programa son variadas y van desde el uso del ingenio por parte del personal de prueba hasta herramientas automatizadas que ayudan a aliviar el peso y el costo de tiempo de esta actividad (Pressman, 2002).

3.3.1 Valoración de la solución

Pruebas funcionales

Las pruebas funcionales tienen como objetivo ejercitar profundamente el sistema comprobando la integración del sistema de información globalmente, verificando el funcionamiento correcto de las interfaces entre los distintos subsistemas que lo componen y con el resto de sistemas de información con los que se comunica (Pressman, 2002). Se seleccionó este tipo de prueba con el objetivo de verificar y valorar las funcionalidades del sistema mediante la especificación de requisitos.

Método de prueba: Caja negra

Como se decide probar el funcionamiento de la interfaz y las funcionalidades del sistema se escoge el método de caja negra. Este método permite comprobar el comportamiento del *software* a través de los requisitos funcionales (Pressman, 2002).

Las pruebas de caja negra pretenden encontrar estos tipos de errores:

- Funciones incorrectas o ausentes.
- Errores en la interfaz.

- Errores en estructuras de datos o en accesos a bases de datos externas.
- Errores de comportamiento o desempeño.
- Errores de inicialización y de terminación.

Técnica de prueba: Partición equivalente

La técnica divide el dominio de entrada de un programa en clases de datos, a partir de las cuales pueden derivarse casos de prueba. Además, descubre clases de errores, que, de otra manera, requeriría la ejecución de muchos casos antes de que se observe el error general. Mediante su empleo se puede reducir al máximo el total de casos de prueba que deben desarrollarse (Cabeza Chávez, 2015).

3.3.2 Resultados de los casos de prueba

Pruebas funcionales

Se realizaron pruebas funcionales sobre la interfaz de *Kibana* con la configuración de *Logstash* incorporada. Se identificaron 9 no conformidades en 3 iteraciones, de estas no conformidades 2 fueron de errores ortográficos y 7 de validación incorrecta.

Se muestra en la Ilustración 9 las no conformidades detectadas en las tres iteraciones realizadas. En una primera iteración se detectaron un total de 3 no conformidades de las cuales fue resuelta 1 y las restantes 2 quedaron pendientes para la segunda iteración, esta su vez arrojó un total de 4 no conformidades más, de las cuales 2 fueron resueltas y 2 quedaron pendientes. Por último, una tercera iteración la cual arrojó un total de 2 no conformidades, ambas terminaron siendo resueltas.

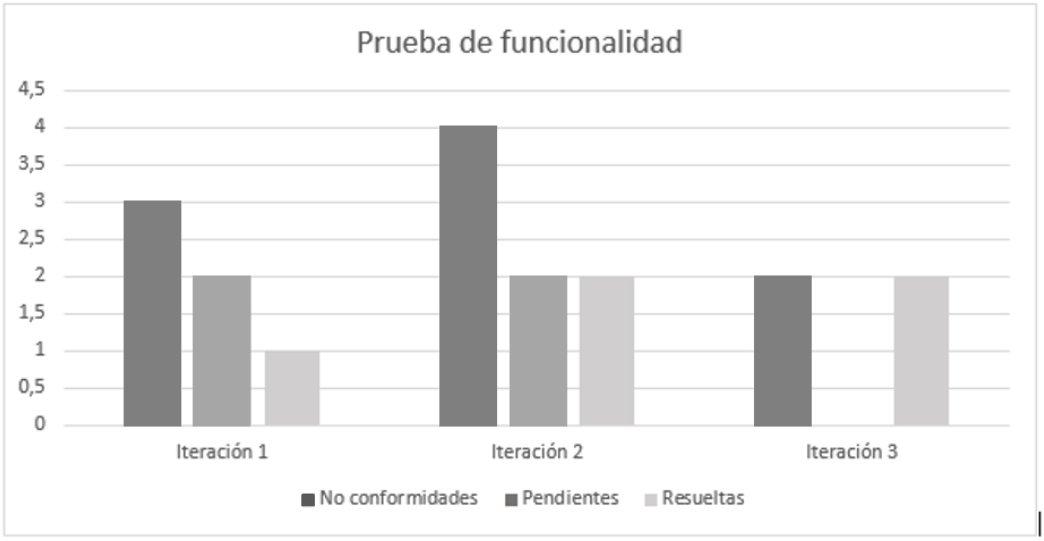


Ilustración 9: Resultados de las pruebas funcionales. Fuente: elaboración propia.

3.4 Pruebas de aceptación

Se decide realizar la prueba de aceptación porque es necesario para la validación de la solución que el cliente este de acuerdo con el funcionamiento del sistema implementado y así pueda emitir una carta de aceptación donde se muestra su conformidad con la solución.

Como técnica se escogen las pruebas alfas, esta se lleva a cabo por un cliente un entorno controlado y se usa el *software* de forma natural con el desarrollador como observador del usuario. Para que tengan validez se debe crear un ambiente con las mismas condiciones que se encontrarán en la instalación final del sistema. Una vez logrado esto se procede a realizar las pruebas y a documentar los resultados.

3.4.1 Técnica IADOV

La técnica de IADOV es utilizada para determinar el nivel de satisfacción individual y grupal de los usuarios a partir de una encuesta elaborada. Dicha encuesta fue aplicada a 5 especialistas como objeto de estudio. A continuación, se muestra la encuesta realizada a los especialistas del centro CESOL.

Encuesta inicial de la investigación

Especialista, le invito a responder el siguiente cuestionario con el fin de conseguir su colaboración en la presente investigación, solicito que exprese en sus respuestas criterios verídicos que guíen al autor de la investigación. Marque con una X en una sola opción y en el caso de la 5 responda brevemente. Por el

tiempo brindado, muchas gracias.

1. ¿Considera importante la implementación de un sistema informático de monitoreo en tiempo real que permita conocer las estadísticas acerca de la instalación de las aplicaciones del repositorio de Nova?

Sí ___ No ___ No sé___

2. ¿Considera usted correcta la forma en que se analizan los *logs* generados por el repositorio de Nova actualmente?

Sí ___ No ___ No sé___

3. ¿Considera usted factible la implementación de un sistema informático de monitoreo en tiempo real que permita conocer las estadísticas acerca de la instalación de las aplicaciones del repositorio de Nova?

Sí ___ No ___ No sé___

4. Luego de haber mostrado los resultados de la solución refleje en qué medida le gusta la solución desarrollada.

__Me gusta mucho __Me disgusta más de lo que me gusta
__Me gusta más de lo que me disgusta __No me gusta nada
__Me da lo mismo __No sé decir

5. ¿Qué opina usted acerca de los beneficios que traería para la universidad disponer de una solución de seguimiento en tiempo real al uso del repositorio de Nova?

La aplicación de esta técnica es una vía indirecta para el estudio de satisfacción, los criterios que se utilizan se fundamentan en las relaciones que se establecen entre las tres preguntas cerradas, que se intercalan dentro de un cuestionario y cuya relación el encuestado desconoce. En este caso, en la encuesta, son las preguntas 2,3 y 4 que se relacionan a través de lo que se denomina el “Cuadro Lógico de IADOV” que se muestra en la siguiente tabla.

Tabla 5: Cuadro lógico de IADOV.

4. Luego de haber mostrado los resultados de la solución refleje en qué medida le gusta la	2. ¿Considera usted correcta la forma en que se analizan los *logs* generados por el repositorio de Nova actualmente?

solución desarrollada.	**No**			**No sé**			**Sí**		
	3. ¿Considera usted factible la implementación de un sistema informático de monitoreo en tiempo real que permita conocer las estadísticas acerca de la instalación de las aplicaciones del repositorio de Nova?								
	Sí	**No sé**	**No**	**Sí**	**No sé**	**No**	**Sí**	**No sé**	**No**
Me gusta mucho	1	2	6	2	2	6	6	6	6
Me gusta más de lo que me disgusta	2	2	3	2	3	3	6	3	6
Me da lo mismo	3	3	3	3	3	3	3	3	3
Me disgusta más de lo que me gusta	6	3	6	3	4	4	3	3	4
No me gusta nada	6	6	6	6	4	4	6	6	5
No sé decir	2	3	6	3	3	3	6	6	4

La forma de utilizar la tabla es la siguiente:

- Cada encuestado recibe una evaluación individual en dependencia de las respuestas que dé a las preguntas cerradas. Para facilitar el procesamiento posterior, en el diseño de la encuesta se debe tener en cuenta que a estas preguntas sólo se responda de la forma prevista en el cuadro lógico de IADOV.

- Las respuestas a las preguntas 2 y 3 pueden ser SI, NO, NO SÉ, y a las preguntas 4, “Me gusta mucho”, “Me gusta más de lo que me disgusta”, “Me da lo mismo”, “Me disgusta más de lo que me gusta”, “No me gusta nada”, o “No sé qué decir”.

Para obtener el índice de satisfacción grupal (ISG) se trabaja con los diferentes niveles de satisfacción que se expresan en la escala numérica que oscila entre +1 y – 1. El número resultante de la interrelación de las tres preguntas indica la posición de cada encuestado en la siguiente escala de satisfacción: clara satisfacción +1, más satisfecho que insatisfecho 0.5, no definido y contradictorio 0, más insatisfecho que satisfecho -0.5 y clara insatisfacción -1.

El índice de satisfacción grupal (ISG) se expresa en una escala numérica que va desde 1 (máxima

satisfacción), hasta -1 (máxima insatisfacción). El ISG se calcula mediante la siguiente fórmula:

$$ISG = \frac{A\,(+1)\ +\ B\,(+0.5)\ +\ C\,(0)\ +\ D\,(-0.5)\ +\ E\,(-1)}{N}$$

En esta fórmula A, B, C, D, E, representan la cantidad de encuestados colocados respectivamente en las posiciones de satisfacción 1; 2; 3 o 6; 4; 5 y donde N representa la cantidad total de encuestados

Por ejemplo, si 4 especialistas presentan máxima satisfacción, 1 están más satisfechos que insatisfechos, el índice de satisfacción se calcularía de la siguiente forma:

$$ISG = \frac{4\,(+1) +\ 1(+0.5)\ +\ 0\,(0)\ +\ 0\,(-0.5)\ +\ 0\,(-1)}{5}$$

ISG=0.7

Ilustración 10: Ubicación del índice de satisfacción grupal. Fuente: elaboración propia.

Los valores de ISG que se encuentran comprendidos entre -1 y -0,5 indican insatisfacción; los comprendidos entre -0,49 y +0,49 evidencian contradicción y los que caen entre 0,5 y 1 indican que existe satisfacción.

Mediante la técnica de IADOV se confirmó la factibilidad de uso del objetivo de la investigación, expresando cuantitativamente en el alto ISG (0.7) y cualitativamente en los criterios emitidos en el centro CESOL, lo que refleja la aceptación de la propuesta y el reconocimiento a su utilidad.

Valoraciones generales del capítulo

- ✓ La utilización de estándares de código para la elaboración de la solución propuesta posibilitó adoptar una estructura homogénea que asegura la calidad de la misma, una menor cantidad

errores y el fácil mantenimiento, así como una mayor comunicación entre el programador y futuros terceros.

- ✓ La detección temprana de errores, mediante la aplicación de pruebas funcionales, que permitió evaluar que el sistema funciona como inicialmente fue diseñado.
- ✓ La prueba de aceptación realizada permitió evaluar la propuesta de solución por parte del cliente y su satisfacción con el producto final.
- ✓ La aplicación de la técnica de IADOV permitió evaluar de manera satisfactoria la propuesta de solución demostrando la satisfacción de los potenciales usuario hacia el sistema desarrollado.

Conclusiones generales

Con la culminación del presente trabajo de diploma se cumplieron cada uno de los objetivos trazados, distinguiéndose de manera general los siguientes aspectos:

- ✓ El análisis y la fundamentación teórico-metodológico de los principales conceptos asociados a la investigación científica, posibilitó la selección de la metodología, herramientas y tecnologías para la implementación de la solución propuesta, las que se enmarcan en AUP en su versión para la UCI como metodología de desarrollo, *Elasticsearch* motor de búsqueda, *Logstash* como conducto de datos, *Kibana* como plataforma para la visualización de datos, *RegExr* como herramienta para construir y probar expresiones regulares y Visual Paradigm para UML como herramienta de modelado.
- ✓ Mediante la generación de reportes con datos estadísticos e información especializada sobre el uso en tiempo real del repositorio de Nova se logró simplificar el proceso de toma de decisiones, mejorando el desempeño y la satisfacción de los usuarios.
- ✓ La aplicación de las pruebas, métodos y técnicas demostraron el correcto funcionamiento del sistema informático de monitoreo en tiempo real del repositorio de Nova, la satisfacción del cliente y los potenciales usuarios hacia la solución desarrollada.

Recomendaciones

Se recomienda:

- Incorporar al repositorio de aplicaciones de Nova las tecnologías *Elasticsearch*, *Logstash* y *Kibana* para futuros proyectos productivos.
- Como la tecnología de visualización *Kibana* está desarrollada en inglés, se recomienda incorporar un paquete de idioma o traducirla al español para una mejor comprensión y utilización de dicha herramienta.

Referencias bibliográficas

Albo Castro, Mónica Ma. 2016. Fundamentos para la definición de un modelo de evaluación de la calidad para GNU/Linux Nova, La Habana : s.n., 2016, Vol. 11, págs. 87-101.

Apache. 2015. [En línea] 2015. [Citado el: 24 de noviembre de 2016.] https://httpd.apache.org/docs/2.0/es/logs.html.

Bahit, E. 2012. Hackers & Developers Magazine. [En línea] 2012. http://www.hdmagazine.org.

Cabeza Chávez, Yanet. 2015. Módulo de JAVA para la herramienta Auditoría de Código Fuente. La Habana : s.n., 2015. Trabajos de diplomas.

Chhajed, Saurabh . 2015. Learning ELK Stack. 2015.

DB-Engines. 2017. DB-Engines [En línea] 2017. [Citado el: 15 de junio de 2017.] https://db-engines.com/en/system/InfluxDB.

Degiorgi, Horacio y Mendez, Adrián. 2011. Aplicación de indicadores para la evaluación de consulta de un repositorio institucional. 2011.

Díaz, J. 2009. Las metodologías ágiles como garantía de calidad del software. 2009. págs. 40-44. Vol. 3.

Elasticsearch. 2015. Getting Started Elasticsearch. [En línea] 2015. [Citado el: 25 de noviembre de 2016.] https://www.elastic.co/guide/en/elasticsearch/reference/current/getting-started.html.

Estévez, A. 2009. Análisis estadísticos. [En línea] 2009. [Citado el: 20 de noviembre de 2016.] http://es.slideshare.net/moagmar/anlisis-estadstico.

Free Software Foundation. 2016. gnu.org. [En línea] Free Software Foundation, 27 de noviembre de 2016. [Citado el: 15 de 11 de 2016.] https://www.gnu.org/philosophy/free-sw.es.html.

Grafana Labs. 2017. Grafana. [En línea] 2017. [Citado el: 22 de Mayo de 2017.] https://grafana.com/.

Graphite. 2017. Graphite. [En línea] 2017. [Citado el: 15 de junio de 2017]. https://graphiteapp.org.

Kibana. 2015. Getting Started with Kibana. [En línea] 2015. [Citado el: 25 de noviembre de 2016.] http://www.elastic.co/webinars/getting-started-kibana.

Logstash. 2015. Getting Started Logstash. [En línea] 2015. [Citado el: 25 de noviembre de 2016.] http://www.elastic.co/webinars/getting-started-logstash.

Mestras. 2013. Servidores Web - Apache. [En línea] Dep. Ingeniería de software e Inteligencia Artificial, 2013. http://www.fdi.ucm.es/profesor/jpavon/web/31-ServidoresWeb-Apache.pdf.

Miranda Domínguez, Laura Elida, y otros. 2014. Repositorio digital. [En línea] junio de 2014. https://repositorio.uci.cu/jspui/handle/ident/8887.

Nova. ¿Qué es Nova? [En línea] 20 de enero de 2015. http://humanos.uci.cu/nova/.

Oficina Nacional De Normalización. 2005. *Norma Cubana.* 2005.

OMG UML. 2007. OMG Unified Modeling Language, Superstructure*.* 2007.

Orellana, L. 2001. Estadística descriptiva. [En línea] 2001. [Citado el: 1 de diciembre de 2016.] http://www.dm.uba.ar/materias/estadistica_Q/2011/1/modulo%20descriptiva.pdf.

Pressman, Roger. 2002. Ingeniería de software, un enfoque práctico*.* 2002.

Pressman, Roger S. 2009. *Software Engineering. A practitioner´s Approach.* New York : McGraw Hill, 2009. ISBN 978-0-07-337597-7.

R. Barton, Mary y M. Waters, **Margaret**. Cómo crear un Repositorio Institucional. Manual LEADIRS II. [Citado el: 20 de mayo de 2017.] http://www.recolecta.net/buscador/documentos/mit.pdf.

RegExr. 2017. [En línea] 2017. https://www.regexr.com/.

Romero, L. 2002. Estadística en la toma de decisiones*.* 2002. pág. 12.

Sneiderman, S y Gómez, J. 2015. Expresión de patologías del desvalimiento a través de técnicas gráficas*.* 2015. págs. 1-30.

Somerville, I. 2005. Ingeniería de Software 7ma edición. [En línea] 2005. [Citado el: 25 de enero de 2017.] http://es.slideshare.net/jasc_584/ingenieriadesoftware-iansommerville7maedicion-9417118.

Stallman, Richar M. 2004. *Software* libre para una sociedad libre*.* Madrid : Traficantes de Sueños, 2004.

Sublime Text. 2017. [En línea] 2017. https://www.sublimetext.com/.

Anexo 1: Historia de usuario

Historia de Usuario	
Número: HU_1	**Nombre:** Añadir *url* de la ubicación del archivo *log* de acceso al servidor *web*.
Prioridad: Alta	**Iteración asignada:** 1
Programador: Raúl Sánchez Arañó	**Tiempo estimado:** 1 día
Riesgo en desarrollo: - Fallo de conexión - Fallo del servidor	**Tiempo real:** 1 día
Descripción: Se define en la configuración de *Logstash* la ruta donde se encuentra ubicado el archivo *log* de acceso al servidor *web*, una vez hecho esto se procede a analizarlo.	
Prototipo: No.	

Historia de Usuario	
Número: HU_2	**Nombre:** Definir la búsqueda a descargas de archivos con extensión .deb
Prioridad: Alta	**Iteración asignada:** 2
Programador: Raúl Sánchez Arañó	**Tiempo estimado:** 1 día
Riesgo en desarrollo: - Fallo de conexión - Fallo del servidor	**Tiempo real:** 1 día

Descripción: Se configurará *Logstash* para que analice las líneas de *logs* buscando descargas de archivos con extensión .deb, o sea, las aplicaciones que se descargan por los usuarios.
Prototipo: ▸ May 23rd 2017, 22:05:41.000 **message:** 10.8.112.187 - - [23/May/2017:22:05:41 -0400] "GET /nova/2015/pool/extendido/7/7kaa/7kaa-dbg_2.14.4-1_amd64.deb HTTP/1.1" 200 5659849 "http://10.8.112.187/nova/2015/pool/extendido/7/7kaa/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:51.0) Gecko/20100101 Firefox/51.0" **@version:** 1 **@timestamp:** May 23rd 2017, 22:05:41.000 **path:** /var/log/apache2/access.log **host:** SAM **clientip:** 10.8.112.187 **ident:** - **auth:** - **timestamp:** 23/May/2017:22:05:41 -0400 **verb:** GET **request:** /nova/2015/pool/extendido/7/7kaa/7kaa-dbg_2.14.4-1_amd64.deb **httpversion:** 1.1 **response:** 200 **bytes:** 5659849 **referrer:** "http ▸ May 23rd 2017, 22:02:31.000 **message:** 10.8.112.187 - - [23/May/2017:22:02:31 -0400] "GET /nova/2015/pool/extendido/7/7kaa/7kaa-data_2.14.4-1_all.deb HTTP/1.1" 200 20248648 "http://10.8.112.187/nova/2015/pool/extendido/7/7kaa/" "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:51.0) Gecko/20100101 Firefox/51.0" **@version:** 1 **@timestamp:** May 23rd 2017, 22:02:31.000 **path:** /var/log/apache2/access.log **host:** SAM **clientip:** 10.8.112.187 **ident:** - **auth:** - **timestamp:** 23/May/2017:22:02:31 -0400 **verb:** GET **request:** /nova/2015/pool/extendido/7/7kaa/7kaa-data_2.14.4-1_all.deb **httpversion:** 1.1 **response:** 200 **bytes:** 20248648 **referrer:** "http

Historia de Usuario	
Número: HU_3	**Nombre:** Mostrar cantidad de descargas por fecha.
Prioridad: Alta	**Iteración asignada:** 3
Programador: Raúl Sánchez Arañó	**Tiempo estimado:** 4 días
Riesgo en desarrollo: - Fallo de conexión - Fallo del servidor	**Tiempo real:** 5 días
Descripción: La visualización de las estadísticas en *Kibana* se harán mediante el campo paqueteDescargado, filtrando una búsqueda de paquetes descargados del repositorio en una fecha determinada.	
Prototipo: -	

Historia de Usuario	
Número: HU_4	**Nombre:** Mostrar las direcciones *IP* que más paquetes descargan del repositorio.
Prioridad: Alta	**Iteración asignada:** 4
Programador: Raúl Sánchez Arañó	**Tiempo estimado:** 4 días
Riesgo en desarrollo: - Fallo de conexión - Fallo del servidor	**Tiempo real:** 5 días
Descripción: Se hará una visualización con gráficas sobre las direcciones *IP* que más paquetes han descargado del repositorio.	
Prototipo:-	

Historia de Usuario	
Número: HU_5	**Nombre:** Mostrar cantidad de descargas por paquetes
Prioridad: Alta	**Iteración asignada:** 5
Programador: Raúl Sánchez Arañó	**Tiempo estimado:** 4 días
Riesgo en desarrollo: - Fallo de conexión - Fallo del servidor	**Tiempo real:** 5 días
Descripción: En las gráficas se visualizará por paquetes cuantas descargas se han realizado	

Prototipo: -

Historia de Usuario	
Número: HU_6	**Nombre:** Mostrar cantidad de paquetes descargados por dirección *IP*.
Prioridad: Alta	**Iteración asignada:** 6
Programador: Raúl Sánchez Arañó	**Tiempo estimado:** 4 días
Riesgo en desarrollo: - Fallo de conexión - Fallo del servidor	**Tiempo real:** 5 días
Descripción: Se mostrarán todos los paquetes del repositorio que fueron descargados desde una misma dirección *IP*	
Prototipo: -	

Historia de Usuario	
Número: HU_7	**Nombre:** Mostrar las 10 primeras direcciones *IP* que más paquetes descargan del repositorio
Prioridad: Media	**Iteración asignada:** 7
Programador: Raúl Sánchez Arañó	**Tiempo estimado:** 2 días
Riesgo en desarrollo: - Fallo de conexión	**Tiempo real:** 1 día

- Fallo del servidor	
Descripción: Mediante un gráfico se visualizarán las 10 primeras *IP* que más paquetes hayan descargado del repositorio	
Prototipo: -	

Historia de Usuario	
Número: HU_8	**Nombre:** Mostrar gráficas de tendencias por semana
Prioridad: Baja	**Iteración asignada:** 8
Programador: Raúl Sánchez Arañó	**Tiempo estimado:** 2 días
Riesgo en desarrollo: - Fallo de conexión - Fallo del servidor	**Tiempo real:** 1 día
Descripción: Se mostrarán gráficas sobre las aplicaciones que se descargaron en una semana dada	
Prototipo: -	

Historia de Usuario	
Número: HU_9	**Nombre:** Mostrar gráficas de tendencias por meses
Prioridad: Baja	**Iteración asignada:** 9
Programador: Raúl Sánchez Arañó	**Tiempo estimado:** 2 días
Riesgo en desarrollo:	**Tiempo real:** 1 día

- Fallo de conexión - Fallo del servidor	
Descripción: Se mostrarán gráficas sobre las aplicaciones que se descargaron en un mes dado	
Prototipo: -	

Historia de Usuario	
Número: HU_10	**Nombre:** Mostrar gráficas de tendencias por años
Prioridad: Baja	**Iteración asignada:** 10
Programador: Raúl Sánchez Arañó	**Tiempo estimado:** 2 días
Riesgo en desarrollo: - Fallo de conexión - Fallo del servidor	**Tiempo real:** 1 día
Descripción: Se mostrarán gráficas sobre las aplicaciones que se descargaron en un año dado	
Prototipo: -	

Anexo 2: Ilustraciones

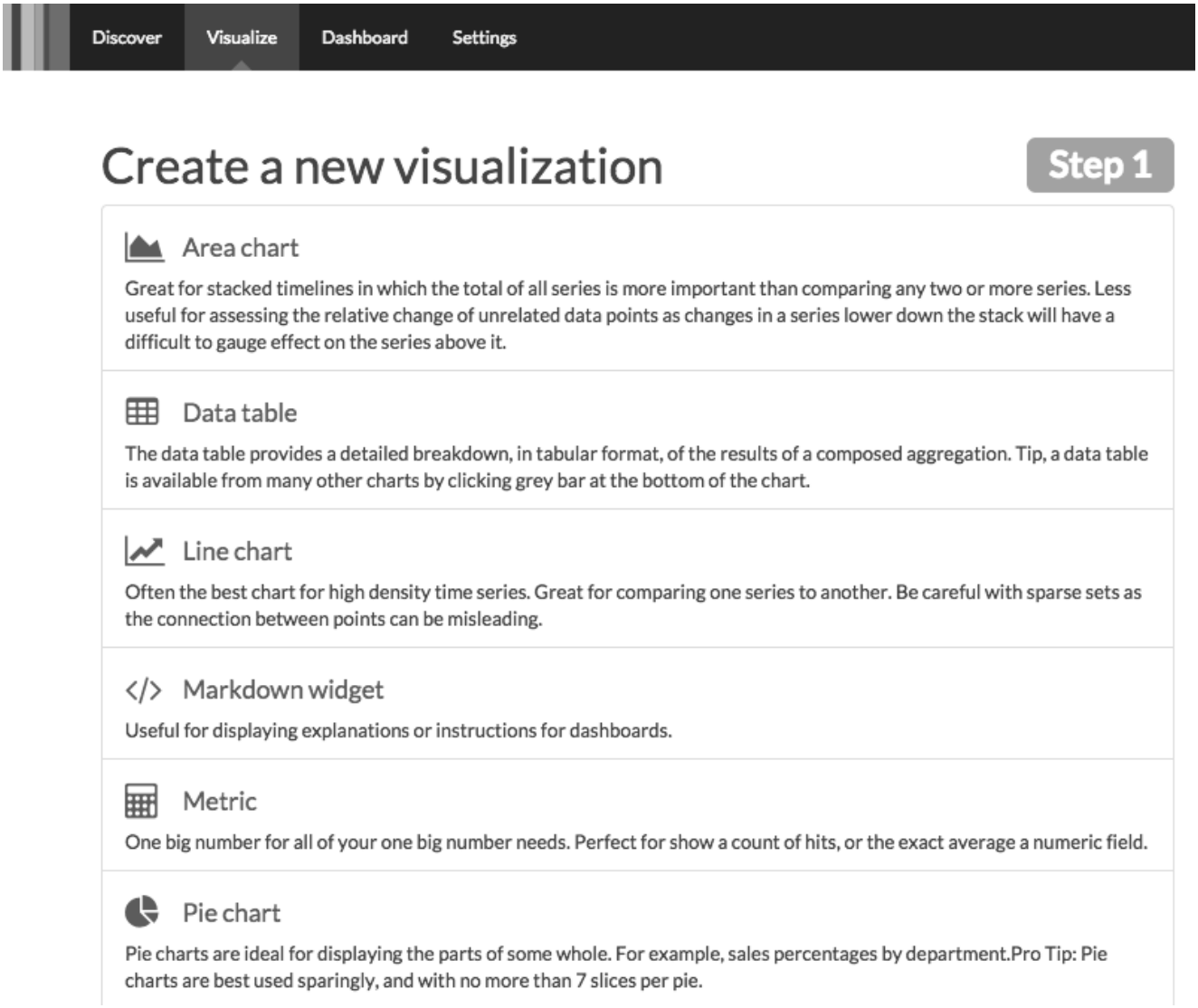


Ilustración 11: Pantalla Visualize de la herramienta Kibana.

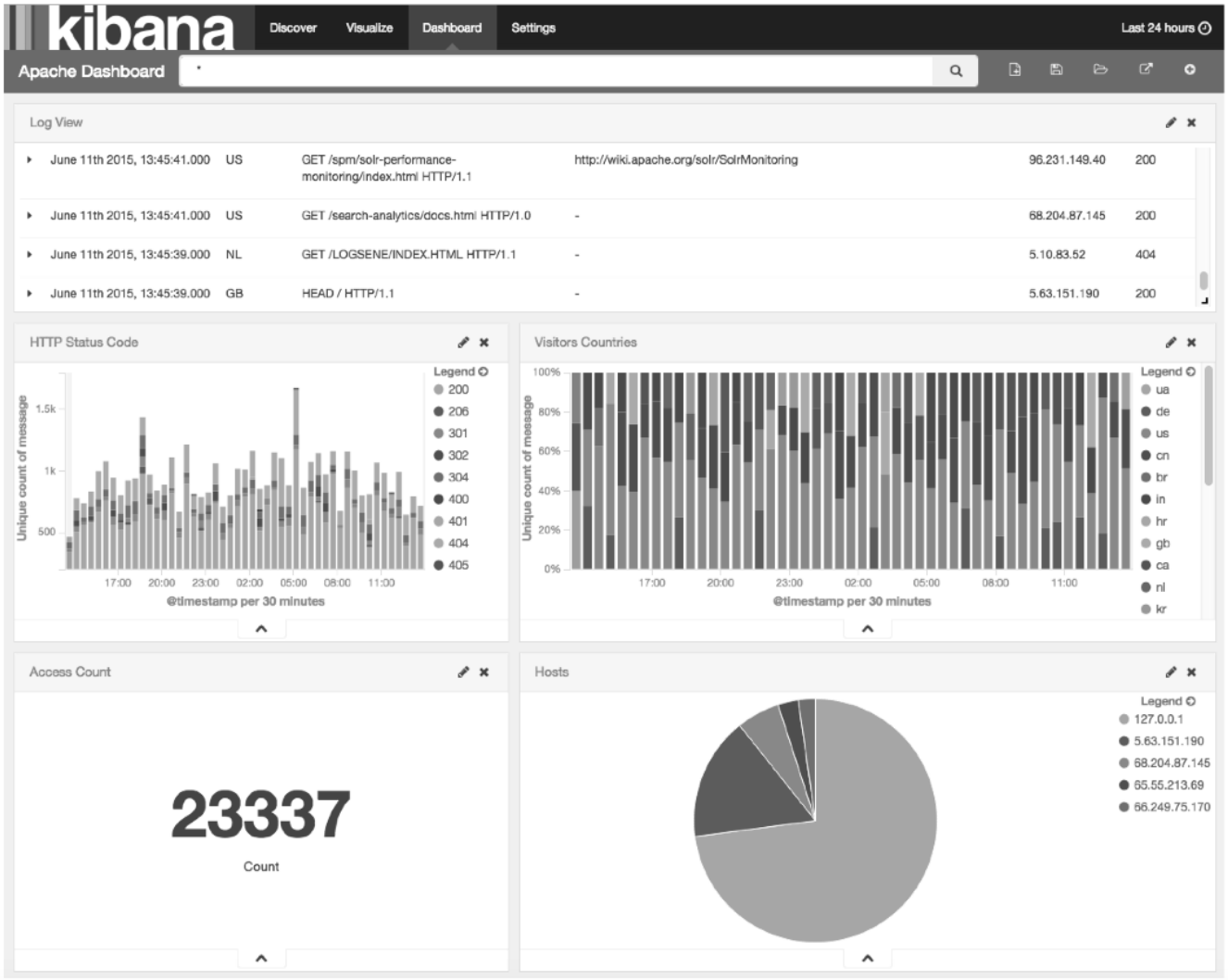


Ilustración 12: Pantalla Dashboard de la herramienta Kibana.

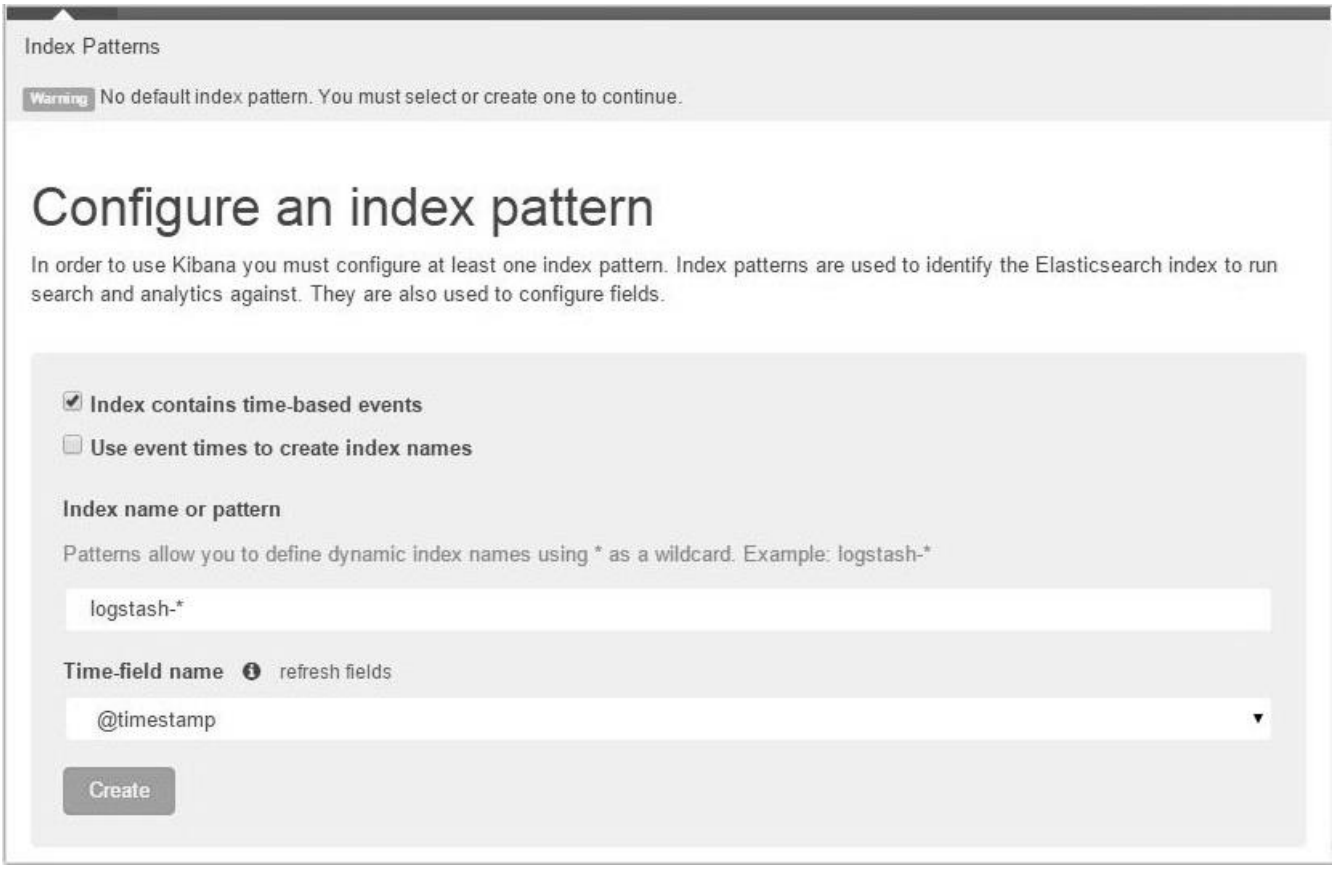


Ilustración 13: Pantalla Settings de la herramienta Kibana.

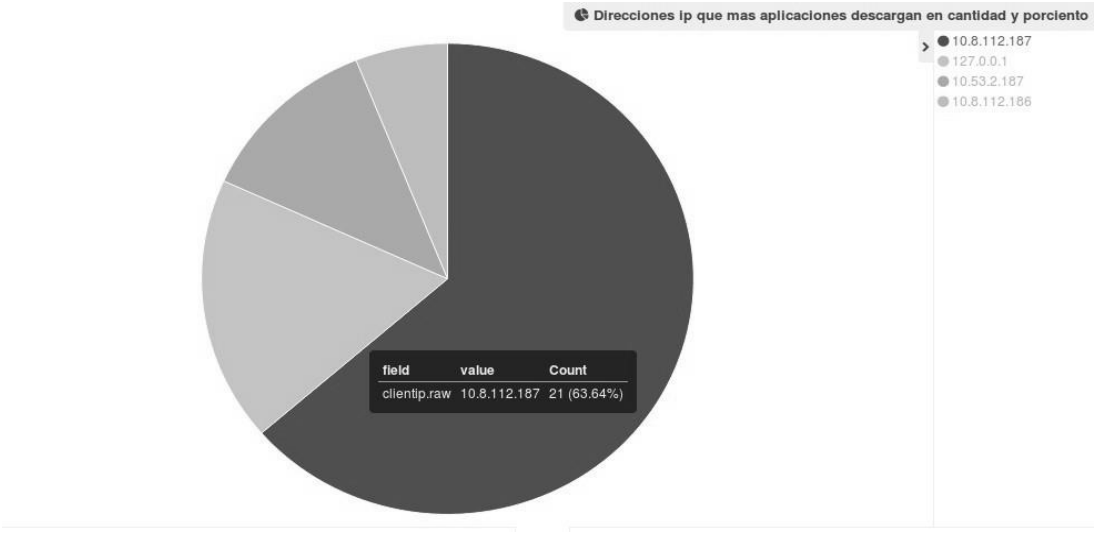


Ilustración 14: Descargas de paquetes por dirección IP.

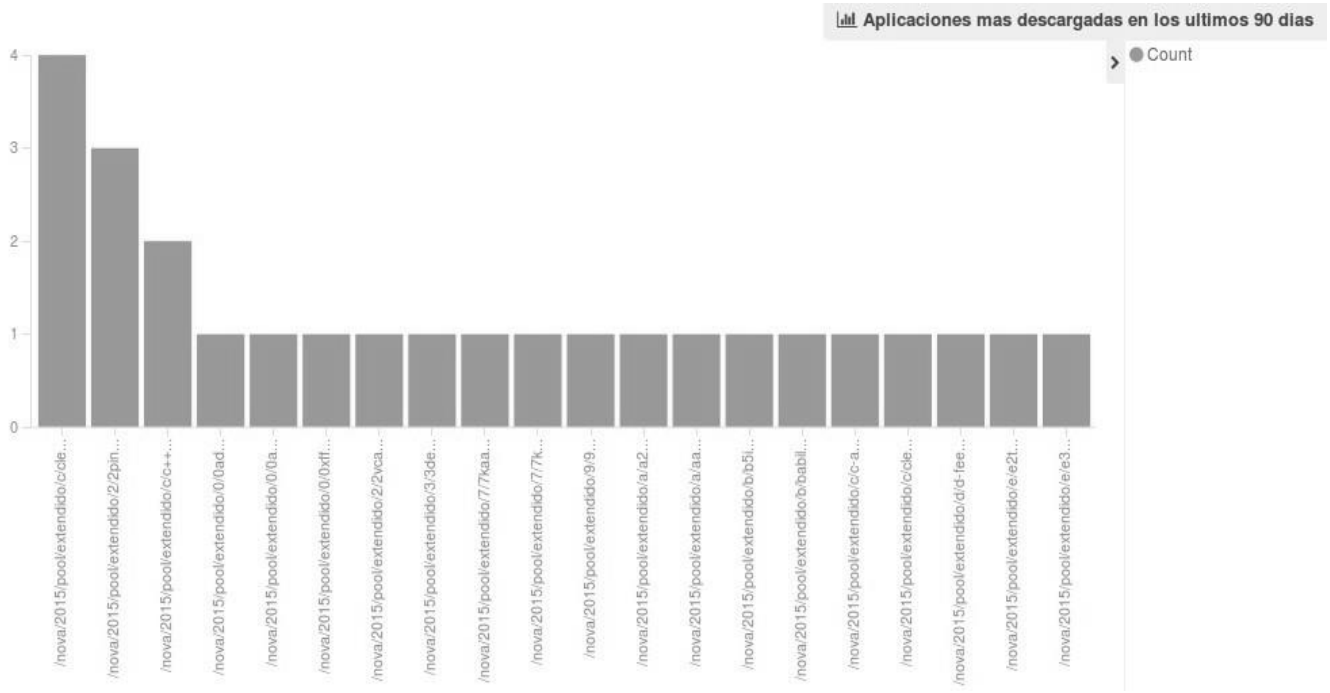


Ilustración 15: Aplicaciones más descargadas en los últimos 90 días.

Glosario de términos

BPMN (*Business Process Model And Notation*): (en español, Notación para el Modelado de Procesos de Negocio), es una forma estándar y gráfica de modelar procesos de negocios.

Código Abierto: término con el que se conoce al *software* distribuido y desarrollado libremente.

GET: Petición para la obtención de datos.

GPL: GNU *General Public License*. Licencia creada por la *Free Software* Foundation en 1989, orientada principalmente a proteger la libre distribución, modificación y uso de *software*. Su propósito es declarar que el *software* cubierto por esta licencia es *software* libre y protegerlo de intentos de apropiación que restrinjan esas libertades a los usuarios.

***Hardware*:** corresponde a todas las partes tangibles de una computadora: sus componentes eléctricos, electrónicos, electromecánicos y mecánicos; sus cables, gabinetes o cajas, periféricos de todo tipo y cualquier otro elemento físico involucrado

HTTP (*HyperText Transfer Protocol*): (en español, Protocolo de transferencia de hipertexto), es un método de intercambio de información en la *World Wide Web* mediante el cual se transfiere el contenido de las páginas *web* a una computadora.

HTML (*Hipertext Markup Language*): utiliza marcas para describir la forma en la que deberían aparecer el texto y los gráficos en un navegador *web* que, a su vez, están preparados para leer esas marcas y mostrar la información en un formato estándar.

IP (*Internet Protocol*): (en español Protocolo de Internet) o *IP* es un protocolo no orientado a conexión usado tanto por el origen como por el destino para la comunicación de datos a través de una red de paquetes conmutados.

OACE (Organismos de la Administración Central del Estado): encargado de trazar, ejecutar y controlar la política del Estado y del Gobierno en relación con la seguridad biológica.

Paquetes: conjunto de ficheros relacionados con una aplicación. Constituyen la forma básica en que las distribuciones del sistema operativo GNU/Linux distribuyen las aplicaciones.

PHP (acrónimo recursivo de PHP: *Hypertext Preprocessor*): es un lenguaje de código abierto muy popular especialmente adecuado para el desarrollo *web* y que puede ser incrustado en *HTML*.

***Plugin*:** (en español: complemento). Es una aplicación que se relaciona con otra para aportarle una función nueva.

RAM (*Random Access Memory*): (en español, Memoria de Acceso Aleatorio), memoria principal de la computadora, donde residen programas y datos, sobre la que se pueden efectuar operaciones de lectura y escritura.

***Software*:** programas del ordenador. Se refiere a instrucciones que se ejecutan por medio de un equipo de cómputo.

URL (*Uniform Resource Locator*): En español, Localizador Uniforme de Recursos. Es la dirección específica que se asigna a cada uno de los recursos disponibles en la red con la finalidad de que estos puedan ser localizados o identificados.

UTC (*Universal Time Coordinated*): zona horaria de referencia respecto a la cual se calculan todas las otras zonas del mundo.