CAMPUS
DE EXCELENCIA
INTERNACIONAL

Graduado en Ingeniería Informática

Universidad Politécnica de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

TRABAJO FIN DE GRADO

Interfaz de visualización de logs en un sistema de información para la gestión de datos en investigación clínica

Autor: Carlos Prado Ventura

Director: David Pérez del Rey

MADRID, JULIO 2016

Tabla de contenido

RESUMEN

Este proyecto se centra en la gestión de logs (o registros) de un sistema de integración de información de ensayos clínicos, por lo que se plantea el desarrollo de una aplicación para visualizar la información de estos logs. Para ello se ha seleccionado y configurado un software conocido como ELK o Elasticsearch Logstash y Kibana [1], con el objetivo de procesar estas unidades de información, de manera continua, almacenarlas y visualizarlas vía Web, para que los usuarios finales puedan discernir claramente los datos destacados, sin necesidad de explorar manualmente los ficheros que contienen estos logs. Y para que estos usuarios puedan instalar, desplegar y mantener el sistema ELK mediante un script de instalación, un script de despliegue y la documentación pertinente.

This project focuses on the management of logs (or records) of an information integration system from clinical trials, which raises the development of an application to display the information in these logs. To this end, it has been selected and configured a software known as ELK or Elasticsearch, Logstash and Kibana [1], in order to process these information units, continuously, to store and to display them through the Web, so end users can discern outstanding data, without having to manually scan the files that contains logs. And so that these users can install, deploy and maintain ELK system using an installation script, a deployment script and relevant documentation.

1. INTRODUCCIÓN

El proyecto que se trata en estas páginas corresponde al apartado de monitorización gráfica de un "Sistema de gestión de datos utilizados en ensayos clínicos".

Este sistema genera una gran cantidad información en forma de logs que se han de revisar de forma manual. Por lo que se puede intuir de esto, dicha revisión es una tarea costosa en tiempo y monótona, porque siempre se ha de revisar la misma disposición de información con variaciones en los datos que se deben destacar. La necesidad de automatizar este proceso y de hacerlo más cómodo para el usuario son las razones del desarrollo del proyecto: una herramienta que de manera automática pueda gestionar toda esta información, tratarla y disponerla visualmente según las necesidades que se tengan.

El documento presente recoge toda la información de desarrollo de esta herramienta desde su fase inicial, como concepto, a su implementación final en un entorno de Producción, detallando las posibles soluciones en respuesta a los objetivos marcados y las posibilidades de mejora.

1.1. Descripción de proyecto

El proyecto de *análisis y visualización de logs (o registros) de un sistema de información para la gestión de datos utilizados en ensayos clínicos*, consiste en definir e implantar una herramienta de visualización de logs, con la capacidad para mostrar gráficamente toda la información requerida, de tal forma que sea menos laboriosa la búsqueda de información clave o errores en el sistema.

La herramienta a implantar no está limitada en arquitectura o en componentes, por lo que la primera fase del trabajo consiste en la búsqueda de información de elementos útiles (frameworks o aplicaciones) y en la definición de una arquitectura que se pueda adaptar a las necesidades del proyecto. Superada esta fase, se debe desarrollar y configurar la mencionada herramienta para cubrir todos los requisitos que se acuerden y los objetivos que se mencionan en el siguiente apartado.

1.2. Objetivos

El objetivo principal de este proyecto es:

1. **El diseño de una interfaz visual**: Para poder mostrar gráficamente los logs del sistema, observar las interrelaciones que hay entre ellos y además de poder realizar búsquedas sobre determinados elementos.

Los objetivos derivados son:

2. **Análisis de los logs**: Por requerimiento de la interfaz visual, los logs deben ser preprocesados para poder realizar búsquedas sobre ellos y relacionar sus componentes.

3. **Almacenamiento de logs procesados**: Análogo al punto anterior. Se requiere guardar toda la información procesada en una base de datos, de tal forma que el frontal pueda obtener la información necesaria para construir las gráficas.

4. **Redactar documentación**: Para el manejo e instalación del software proporcionado.

En resumen: la meta principal del proyecto, que es la visualización de los logs, se considera el objetivo principal a cubrir. De éste derivan otras necesidades a abarcar, como son, el análisis y almacenado de los logs para poder obtener la información y transmitirla al componente que se encarge de visualizar, y la generación de una documentación que permita exportar, instalar y administrar todo el software entregado.

1.3. Solución propuesta

En este apartado se define arquitectura del sistema que cubre los objetivos mencionados en el anterior apartado, identificando cada componente y el flujo de la comunicación.

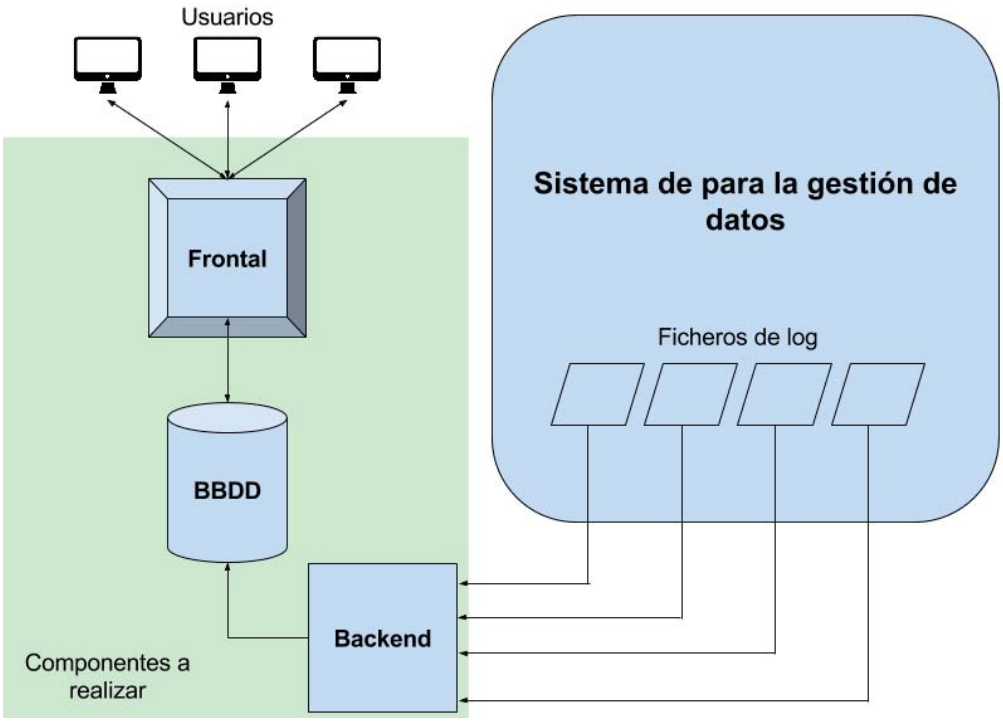

Imagen 1 - Arquitectura propuesta

Como se puede observar en la imagen anterior, ver "Imagen 1 – Arquitectura propuesta", para disponer de un frontal o un elemento visualizador es necesario primero preprocesar en un backend la información de los logs y guardarla en una base de datos. Por lo tanto, para poder satisfacer las necesidades son necesarios, al menos, tres elementos:

1. Sistema que procese los logs.
2. Base de datos.
3. Frontal de visualización.

A partir de este punto se puede comenzar a hacer desarrollos para crear diseñar y crear los distintos componentes desde cero. No obstante, ya existen en Internet herramientas que está optimizadas para realizar labores de revisión o visualización de logs, que pueden ayudar a simplificar esta arquitectura de componentes.

Para este proyecto se destacó ELK, por tener la capacidad de cubrir completamente la arquitectura planteada.

2. TECNOLOGÍAS

En este capítulo se definen y especifican todas las tecnologías que se han usado para dotar de la funcionalidad necesaria al proyecto con el fin de cumplir sus objetivos y requisitos.

2.1. ELK

> The Elastic Stack — that's Elasticsearch, Logstash, Kibana, and Beats — are open source projects that help you take data from any source, any format and search, analyze, and visualize it in real time. (Elastic, 2016) [2]

Como se indica en la referencia anterior, ELK es principalmente el conjunto de tres herramientas: Elasticsearch, Logstash y Kibana (Beats no se ha utilizado en este proyecto) que se utilizan de forma conjunta para gestionar, almacenar y representar datos en tiempo real.

ELK está pensado principalmente para ser usado en sistemas grandes, con mucho flujo de información constante, para Big Data por ejemplo. Por lo que no supone ningún problema para este proyecto, ya que el flujo de información será menor al gigabyte diario.

En los siguientes apartados se detalla cada componente del stack de ELK.

2.2. Elasticsearch

> Elasticsearch is a distributed, open source search and analytics engine, designed for horizontal scalability, reliability, and easy management. It combines the speed of search with the power of analytics via a sophisticated, developer-friendly query language covering structured, unstructured, and time-series data. (Elastic, 2016) [2]

Adicionalmente a la información referenciada, Elasticsearch se basa en Apache Lucene, un API de código abierto encargado de la búsqueda y el indexado de datos, muy utilizado por los motores de búsqueda en Internet.

Elasticsearch es capaz de comunicarse mediante su interfaz web RESTful con otros sistemas y además guarda la información en documentos formato JSON, puesto que también actúa como una base de datos no-sql. [3,4]

En el proyecto: Ésta herramienta cubre las necesidades de procesar los datos, de almacenarlos y de transmitirlos a otros componentes.

2.3. Logsasth

> Logstash is a flexible, open source data collection, enrichment, and transportation pipeline. With connectors to common infrastructure for easy integration, Logstash is designed to efficiently process a growing list of log, event, and unstructured data sources for distribution into a variety of outputs, including Elasticsearch. (Elastic, 2016) [2]

Esta herramienta se trata de un pipeline, 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) y salida u output (por donde sale).

Cada una de estas partes tiene a su disposición varios plugins que, en el caso de la parte de entrada, permiten obtener datos de distintas fuentes, por ejemplo: ficheros, puertos o bases de datos. En el caso de salida, permiten transmitirlos a otra gran variedad de destinos, por ejemplo: Elasticsearch, bases de datos o salida estándar. Y en el caso de filtro permiten definir, alterar o analizar la información, por ejemplo: Grok, multiline o Ruby. [5]

En el proyecto: Logstash realiza la función de obtención, análisis, modificación y reenvío de información a Elasticsearch.

2.4. Kibana

> Kibana is an open source data visualization platform that allows you to interact with your data through stunning, powerful graphics. From histograms to geomaps, Kibana brings your data to life with visuals that can be combined into custom dashboards that help you share insights from your data far and wide. (Elastic, 2016) [2]

Finalmente, Kibana trata todo el apartado de visualización y manejo de información que hay en Elasticsearch que, como se indica en la anterior referencia, es capaz de mostrar distintos diagramas de visualización, configurar uno o varios dashboard con la información que requieran los usuarios, permite exportar los resultados y la capacidad de realizar búsquedas y filtros sobre los datos guardados en Elasticsearch. [6]

En el proyecto: Kibana permite la visualización y monitorización de las unidades de información procesadas.

3. REQUISITOS DEL PROYECTO

La siguiente lista de requisitos son los indicados durante el desarrollo del proyecto, en este apartado se enunciarán y se describirán. En el apartado 4. DISEÑO E IMPLEMENTACIÓN se especificará el desarrollo de los requisitos.

Los requisitos dados son:

1. **Sustitución de número de conceptos por su nombre o descripción:** Los conceptos en los logs son identificadores numéricos que requieren ser transformados en los nombres o descripciones correspondientes de SNOMED [7].

2. **Gráficas de conceptos "Original" y "Expanded":** Visualizaciones en histogramas de la cantidad de conceptos que hay en los logs por fecha.

3. **Indexado del número de pacientes:** se requiere enriquecer los logs de query_llog_GBG.txt para añadir el número de pacientes en los resultados.

4. **Utilización del timestamp definido por los logs:** El timestamp por defecto se establece en el orden de llegada de los logs a Logstash, pero se requiere que éste timestamp se establezca con la fecha definida por los logs.

5. **Relación gráfica de número de pacientes con los conceptos:** Visualización que compare el número de pacientes con los conceptos mencionados en los puntos anteriores.

6. **Distinción de conceptos de los logs tipo "Templates":** Los conceptos obtenidos de los logs del fichero "queryBuilder.log" se deben definir con otro índice distintivo.

7. **Mostrar datos de un concepto en concreto al hacer click en una gráfica:** Obtener más información en las visualizaciones.

8. **En los log tipo Templates, idexar los métodos con sus atributos:** Las líneas de logs que indiquen métodos y atributos deben ser obtenidas en Logstash como un solo bloque de información con todos los índices necesarios.

9. **Sustitución del campo de resultado en query_llog_GBG.txt**: se requiere modificar la línea "%%%%%Works!" por "Operation Successful".

10. **Gráficas, tipo "Top 50", de los conceptos por su nombre o descripción:** Visualizaciones con los nombres o descripciones de los conceptos y no con su identificador.

11. **Gráfica de media de tiempos de ejecución:** Visualización con el tiempo medio de ejecución de las operaciones por timestamp. El tiempo de ejecución de cada operación se incluye en los logs y se debe indexar en ELK.

12. **Mejorar la cantidad de datos:** Aumentar la cantidad de datos provistos en las visualizaciones.

13. **Realizar scripts de instalación y despliegue:** Creación de dos scripts automáticos que, en su ejecución, realicen estas labores de instalación de ELK y los componentes necesarios para su uso, y de despliegue de los distintos servicios de Elasticsearch, Kibana y Logstash.

Todos los anteriores requisitos se han podido implementar y cerrar adecuadamente salvo los siguientes:

Requisito 5: Por limitación de Elasticsearch. No se ha sido posible representar ambos valores en una misma visualización por pertenecer a índices de logs distintos (Elasticsearh no puede relacionar las búsquedas). Se puede solucionar, pero requiere cambios en los ficheros de logs (no especificado) para unir ambos logs y poder realizar un índice compartido de conceptos y pacientes.

Requisito 7: Por limitación de Kibana. Al hacer click sobre un concepto o un dato en una visualización se delimita más el marco de tiempos y se centra en las fechas donde ocurrió ese log. Aunque es posible obtener más información de los logs en las visualizaciones haciendo click en el icono con forma de flecha en la parte central inferior de la gráfica mostrada.

4. DISEÑO E IMPLEMENTACIÓN

Finalmente la arquitectura completa estará definida por cada componente del ELK:

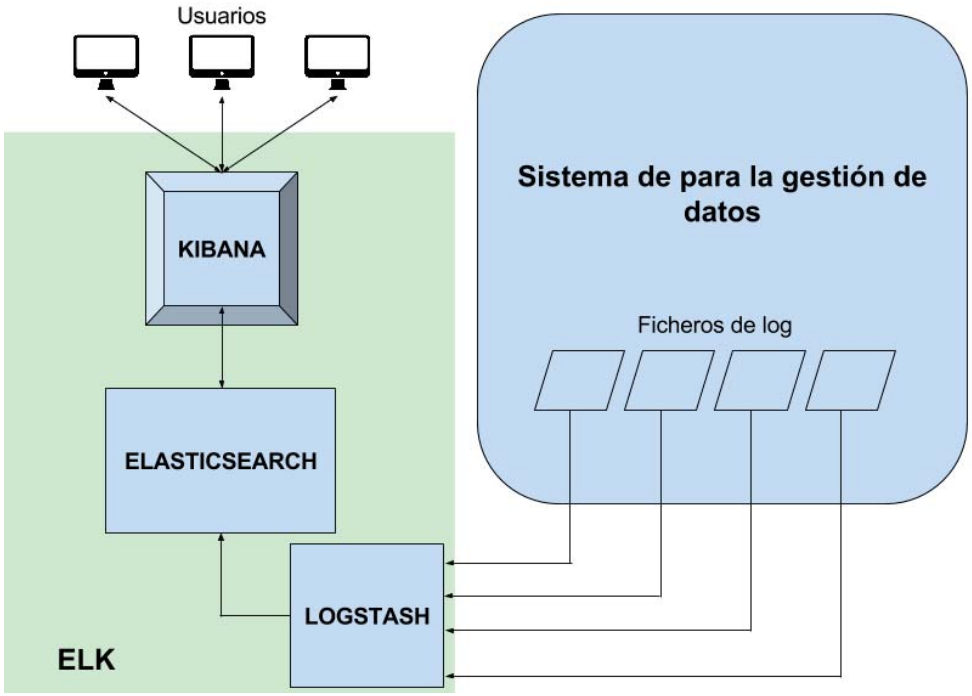

Imagen 2 - Arquitectura final de ELK

Como se puede comprobar todos los componentes de esta gráfica encajan con los descritos en la gráfica del punto 1.3 SOLUCIÓN PROPUESTA (ver "Imagen 1 – Arquitectura propuesta"): Kibana actúa como frontend, mientras que Elasticsearch y Logstash actúan como backend.

La configuración se detalla en los manuales de instalación y de usuario anexados. Este capítulo se centrará en explicar el funcionamiento de ELK para cumplir con los objetivos, explicando lo necesario de la configuración.

4.1. 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 como va a tratar cada línea de logs.

En concreto la configuración realizada en este proyecto se encuentra en el archivo "ConfigBase.conf" (anexado, "Fichero de configuración de Logstash"), que se explicará en detalle a continuación.

4.1.1. Entrada de información

Los logs entran al 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. En este proyecto esta ejecución en múltiples hilos está impedida por el plugin "multiline", que forma parte del filtro (se comentará en el siguiente punto 4.1.2).

La entrada de información está definida por el parámetro "input {}". Dado que la fuente información especificada son ficheros de logs, la entrada de datos está orientada a ficheros, por lo que se utiliza el plugin "file{}".

En el plugin de file se ha de indicar:

- **Type**: identificador del tipo que se va a procesar. Hay tres tipos definidos en el proyecto:
  - Tipo A: Logs que provienen del fichero "autocomplete.txt".
  - Tipo B: Logs que provienen de los ficheros "query_log_GBG.txt" y "query_log_fake.txt". Tienen el mismo tipo porque ambos ficheros contienen logs con el mismo formato.
  - Tipo C: Logs que provienen del fichero "queryBuilder.log".

- **Path**: ruta completa al fichero del tipo correspondiente.

- **Start_position**: desde dónde se requiere comenzar el análisis del fichero. Hay dos opciones:
  - Beginning: desde el principio del fichero.
  - End: desde el final.

En todos los casos se ha elegido como posición inicial “desde el principio del fichero”, para poder trabajar con los datos de los ficheros desde el arranque del Logastash.

En la primera construcción también se encuentra otro parámetro con valor: sincedb_path => “/dev/null”. Este, como se indica en capítulo 3 del manual de usuario del anexo 10.6, se trata de un parámetro que indica donde está el archivo que tiene el puntero de lectura en los ficheros de log. Si se le indica de esta forma “/dev/null” que no hay archivo de puntero, Logstash vuelve a analizar los ficheros.

4.1.2. Procesamiento de los logs

El procesamiento se realiza en el plugin de "filter{}", para poder extraer y diferenciar los elementos destacados de los logs, además de hacer modificaciones sobre ellos, con la finalidad de visualizarlos en Kibana.

La configuración para cada tipo de log es la siguiente:

4.1.2.1. Logs de tipo A

En primer lugar, con el plugin de "multiline" se juntan las líneas de log como una única unidad cada vez que se reciba el carácter '[' en primer lugar, hasta el log que empiece por el mismo carácter. La utilización de este plugin supone no poder analizar los logs en varios hilos, no soporta la ejecución multi-hilo (multithreading): dos líneas de logs contiguas podrían ser separadas en dos hilos y se perdería la posibilidad de combinarlas en una sola unidad de información. Por este motivo, Logstash trabaja con un solo hilo.

Suponiendo que Logstash lee el siguiente contenido del fichero:

```
[Fri Sep 18 14:34:44 CEST 2015] Calling getSuggestionsByFilters(ca,20,Entity,)
[Fri Sep 18 14:34:44 CEST 2015] Executed query:
SELECT autocomplete.id as code, autocomplete_synonyms.label as value, autocomplete.fs_code as fs_code, autocomplete.fs_label as fs_label, autocomplete.root as root, autocomplete.voc as vocid FROM autocomplete, autocomplete_synonyms WHERE autocomplete_synonyms.label LIKE '%ca%' AND root='Entity' AND autocomplete_synonyms.id=autocomplete.id GROUP BY `code` ORDER BY vocid DESC, LENGTH(`value`) ASC, `value` LIMIT 20
…
```

Tras pasar por el multiline se obtendría lo siguiente:

```
[Fri Sep 18 14:34:44 CEST 2015] Calling getSuggestionsByFilters(ca,20,Entity,)
```

En los siguientes procesos se trata de extraer la información destacada (la fecha, el método y sus argumentos).

En segundo lugar está el análisis y estructurado de la información recibida por el plugin de "grock", en el cual se detalla mediante una sentencia en regrex (expresión regular). En este caso, la estructura está definida por "[fecha] método(argumentos)".

La fecha y el método es común para todos los logs y la parte que varía son los argumentos, por lo que en esta primera fase de Grok se obtienen los dos primeros valores.

```
fecha: [Fri Sep 18 14:34:44 CEST 2015]
evento: Calling getSuggestionsByFilters
resto: (ca,20,Entity,)
```

Por otro lado, el plugin "mutate" se encarga de modificar el log para dejarlo libre de caracteres o componentes no necesarios. Tras aplicar el mutate:

```
fecha: Fri Sep 18 14:34:44 2015
evento: Calling getSuggestionsByFilters
resto: (ca,20,Entity,)
```

Uno de los requisitos (requisito 3) es el de sustituir el timestamp por el indicado por el log. De esto se encarga el plugin "date":

```
@timestamp = September 18th 2015, 14:34:44
```

Finalmente, mediante condicionales se identifica el evento (método), para guardar el índice de los argumentos. En este fichero hay tres tipos de logs:

- Executed query: en este caso el argumento es una query y se indexa con este nombre "query".
- Calling getSuggestionsByFilters: indexa los argumentos como "Attributes". Éste es el caso del ejemplo.
- Calling getMetaData: indexa los argumentos como "Attributes".

En el ejemplo, los campos destacados, quedaría de la siguiente forma:

```
@timestampt: September 18th 2015, 14:34:44
EventType: Calling getSuggestionsByFilters
Attributes: (ca,20,Entity,)
```

4.1.2.2. Logs de tipo B

Este tipo tiene dos ficheros de procedencia de log, a diferencia de los otros dos tipos, porque estos ficheros tienen la misma estructura de logs, lo que permite la construcción del mismo analizador.

En primer lugar, crea conjuntos de líneas de logs con el plugin multiline, pero difiere del apartado anterior: busca nuevas líneas que comiencen con el parámetro del nombre del día del log (Mon, Tue, Wed, Thu, Fri, Sat o Sun) y acumula todas las líneas de logs siguientes hasta encontrar otro parámetro con el nombre del día.

En un ejemplo, tomando un fragmento del bloque de log completo:

```
Thu Sep 17 13:43:07 CEST 2015 - [LOG - EXPANDED QUERY]
-----------------------------------
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX hl7rim: <http://hl7rim.GIB-UPM.org/common-data-model#>
PREFIX owl: http://www.w3.org/2002/07/owl#
…
```

En segundo lugar, grok analiza la estructura para encontrar la fecha completa, el método y el resto del mensaje:

```
fecha: Thu Sep 17 13:43:07 CEST 2015
método: [LOG - EXPANDED QUERY]
resto del mensaje:
-----------------------------------
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX hl7rim: <http://hl7rim.GIB-UPM.org/common-data-model#>
PREFIX owl: http://www.w3.org/2002/07/owl#
…
```

Para pasar la fecha a @timestamp se realizan las mismas operaciones que en el apartado anterior y, de la misma manera, se modifica el contenido del bloque de información para facilitar las siguientes operaciones de análisis y para cambiar la línea “%%%%%Works!” por “Operation Successful”, como se comentará en el punto de [LOG – RESULT] (ver más adelante).

En tercer lugar se realiza un análisis del contenido del bloque según el método que corresponda:

- [LOG - EXPANDED QUERY]
- [LOG - GIVEN QUERY]
- [LOG - SQL QUERY]
- [LOG - RESULT]

Teniendo en cuenta los siguientes fragmentos del log:

```
…
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX rev: http://purl.org/stuff/rev#
…
SELECT DISTINCT ?id ?code ?patientId ?birthTime ?effectiveTime_start
?effectiveTime_end ?entityId ?entityClassCode ?entityDeterminerCode ?entityCode
?entityTitle
WHERE {
…
FILTER (?entityCode != '337915000')
FILTER (?entityCode IN ('372540003', '108788001', '108790000',
…
```

Los dos primeros puntos son muy similares porque utilizan el plugin de Ruby, para realizar las siguientes operaciones:

1. Guardar como índices todos los valores tipo “PREFIX” (ver el recuadro del ejemplo anterior).
2. Realizar una búsqueda en el contenido para extraer e indexar en un array todos los conceptos numéricos que se encuentren.
3. Sobre el array de conceptos encontrados, mediante un diccionario, se identifica cada concepto con su nombre o descripción de SNOMED y se guarda en un nuevo índice.

Tras esto se procesa todo el resto del mensaje como un índice “query”.

Del ejemplo se obtiene lo siguiente:

```
…
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX rev: http://purl.org/stuff/rev#
…
```

ConceptsExpanded: 337915000, 372540003, 108788001, 108790000 …
ConceptsExpandedByName: "Idarubicin hydrochloride", "Daunorubicin citrate", …
Query: SELECT DISTINCT ?id ?code ?patientId ?birthTime ?effectiveTime_start …

En [LOG - SQL QUERY] se procesa el resto del mensaje como una query, al igual que lo anterior.

Por último, en [LOG - RESULT] lo más destacado es el estado, que se procesa como un índice "Status" (de ahí la modificación realizada sobre "%%%%%Works!"); el tiempo de ejecución (en milisegundos) y el número de pacientes, por lo que ambos se procesan en dos índices, de valor entero, distintos.

Por ejemplo:

Status: Operation Successful
ExecutionTime_int: 1522
Patients_int: 15

4.1.2.3. Logs de tipo C

Al igual que los tipos anteriores, la primera operación que se realiza es la de agrupar las líneas de logs en conjuntos (en este caso por el carácter '-').

Este es un fragmento del bloque de ejemplo:

```
--------------------------------------------------------------------------
INFO    2015-09-21 10:44:56,173 [http-bio-8443-exec-822]
QueryBuilder.QueryBuilderService  - [getUncontextualizedTemplate]
INFO    2015-09-21 10:44:56,176 [http-bio-8443-exec-822]
QueryBuilder.QueryBuilderService  - [CONCEPT GIVEN] - 11535-2
INFO    2015-09-21 10:44:56,189 [http-bio-8443-exec-822]
QueryBuilder.QueryBuilderService  - [gT-binding] – Observation
…
```

A diferencia de los anteriores, estos bloques poseen varias fechas, de la cual se tomará como @timestamp la primera (la línea que define la operación).

Hay tres posibles operaciones:

- [getContextualizedTemplate]
- [getUncontextualizedTemplate]
- [Result getTemplateAuto]

En los primeros puntos se realiza la misma operación con el plugin de Ruby y se destacan los valores a indexar:

- La fecha
- La URI del servicio
- El nombre del servicio
- La operación

Previamente, se eliminan los caracteres no necesarios y los tres primeros puntos que se repiten en todas las líneas del bloque de información. Sobre el ejemplo:

@timestamp: September 21th 2015, 10:44:56,173
ExecUri: QueryBuilder.QueryBuilderService
QueryBuilderService: QueryBuilder.QueryBuilderService
Operation: [getContextualizedTemplate]
Resto:
[CONCEPT GIVEN] - 11535-2
[gT-binding] – Observation …

El resto de los parámetros son procesados por el plugin de Ruby, como índices independientes:

CONCEPT GIVEN: 11535-2
gT-binding: Observation

Para el último punto de las operaciones ([Result getTemplateAuto]), se procesa y se crea un índice con la query, eliminando toda aquella información no necesaria con el plugin de mutate.

Tras terminar con todas los procesos, el apartado de filtro termina con el borrado de algunos de los índices creados durante el proceso que, o bien, no son necesarios o contienen información redundante.

4.1.3. Salida de la información

La salida está definida por "output{}", donde se utilizan dos plugins:

- Elasticsearch: para la salida de los datos procesados a su servidor, cuya localización se indica en el atributo "hosts". En este caso Elasticsearch se localiza en localhost.

- Stdout: salida estadar, utilizada en las primeras pruebas y en las pruebas con los scripts de Ruby mencionados en el apartado anterior.
  La salida del log de debug de Ruby se indica en el atributo "códec" y es "rubydebug".

Además de éste fichero de configuración, para sus procesos, Logstash requiere de otros dos archivos:

- **Archivo de patrones**: en el que se definen más patrones que los definidos por Grok, para su uso en el fichero de configuración (anexado, "Fichero de patrones").

- **Diccionario de conceptos**: relación de conceptos numéricos con su correspondiente nombre o descripción, para su uso en el plugin de Ruby en el fichero de configuración del Logstash.

4.2. Configuración del Elasticsearch

La ventaja de usar Elasticsearch es que no se requiere de configuración adicional para este proyecto. Simplemente, se ha de tener en cuenta que la configuración sea correcta en el fichero "elasticsearch.yml":

- Host: Localhost
- Port: 9200 (por defecto)

4.3. Configuración de Kibana

La configuración se realiza mediante el fichero "kibana.yml", lo más relevante es:

- Port: 5601 (por defecto)
- Host: 0.0.0.0 (para cualquier conexión de entrada)
- Host de Elasticsearch: http://localhost:9200 (host de conexión con el servicio).

Todo lo relativo a Kibana se explica en el anexo: "Manual de Usuario de ELK".

En este apartado se destaca el objetivo principal del proyecto, la visualización y monitorización de los logs:

4.3.1. Visualizaciones

Las gráficas se componen en su apartado de "Visualize", a raíz de las búsquedas que se hagan 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.

4.3.2. Dashboard

Apartado que permite disponer las gráficas sobre la interfaz web, pudiendo ser guardada dicha configuración y cargada según las necesidades.

4.4. Script de instalación

Se trata de un script de BASH que configura el entorno (Xubuntu) e instala el ELK, con comentarios en inglés para ayudar a su entendimiento. Las acciones que realiza son:

1. Instalación del software "Unzip", para el descomprimir.
2. Creación de la carpeta "ELK" en el usuario indicado.
3. Descompresión de los archivos comprimidos de Elasticsearch, Logtash y Kibana sobre la carpeta de ELK.
4. Creación de la carpeta "jvm" en /usr/lib/ para la instalación de Java 1.8.
5. Descompresión del archivo comprimido de Java 1.8 en la carpeta jvm.
6. Asignación de enlaces simbólicos, permisos y remplazo de la versión de java de la máquina por la 1.8.
7. Copiado de todos los archivos de configuración de ELK y Java 1.8 en sus respectivas carpetas.
8. Cambio de permisos y propiedad para el usuario indicado de la carpeta de ELK.

El script se detalla en el anexo: "Documento de instalación de ELK".

4.5. Script de despliegue

Script en BASH análogo a lo anterior para desplegar automáticamente cada herramienta de ELK. Las acciones que realiza son:

1. Apertura la sesión "elk_elastic" en "tmux" y despliegue de Elasticsearch.
2. Tras 60 segundos, para permitir el despliegue del punto anterior, se realiza el despliegue de Kibana en la sesión "elk_kibana".
3. Tras 30 segundos de espera realiza el despliegue de Logstash en la sesión "elk_logstash".

Logstash está configurado para analizar los ficheros de los logs desde el momento de arranque. Tras un tiempo (10 minutos aproximadamente) ELK tendrá toda la información necesaria para poder trabajar sobre la información dada.

El script se detalla en el anexo: "Documento de instalación de ELK".

5. PROBLEMAS ENCONTRADOS Y SOLUCIONES

A continuación se listan los problemas en el desarrollo de los requisitos y las pruebas, que han ocasionado retrasos en la elaboración de este proyecto, así como algunas soluciones encontradas.

- **Sustitución de timestamp por la fecha proporcionada por los logs**: desde Elastic no se recomienda este cambio, por poder causar inconsistencias a la información.
  Sin embargo, se podía hacer la sustitución con el plugin de date, según se muestra en el post de Stackoverflow de la referencia [8].
  Tras realizar esto, se observó que las fechas no eran correctas, ya que no se resolvía adecuadamente la zona horaria. En otro post, en los foros de soporte de Elastic (ver referencia [9]), este problema se resolvía al añadir el campo "timezone=>"UTC"" que no modificaba la zona horaria (se establecía a +00:00).

- **Uso de diccionario para traducir los conceptos numéricos a nombres**: en Logastash no hay ningún plugin que ayude directamente a hacer esta tarea, por lo que se optó por la realización de un script en Ruby. Para ello se observaron diversos ejemplos de uso de Ruby en la web como, por ejemplo, el utilizado para extraer un patrón en un texto (ver referencia [10]).

- **Instalación de Java 1.8:** en versiones anteriores a la actual del script de instalación se utilizaba el repositorio de Oracle para la instalación de Java 1.8, pero la máquina de producción del departamento no tiene acceso a este repositorio, por ello, se usó el fichero comprimido de java para la instalación. El segundo problema surgió cuando en dicha máquina también estaba instalada la versión 1.7 de Java y debía ser sustituida. Para esto se utilizó lo indicado en la referencia [11].

- **Reindexado de la información:** durante la etapa de validación fue necesario eliminar todos los índices de Elasticsearch y eliminar los punteros de Logstash en los ficheros de logs, para que este último los vuelva a analizar.
  Esto se realizó en primera instancia con el parámetro "sincedb_path => "/dev/null"" (ver referencia [12]). Sin embargo, daba problemas: en ciertas ocasiones, Logstash no analizaba los archivos, por lo que se optó por localizar los ficheros que utilizaba Logstash como puntero y borrarlos.

En resumen: se han encontrado cuatro grandes dificultades durante el desarrollo de este proyecto, las dos primeras son relativas a la configuración de Logstash (problemas con el timestamp y los diccionarios), la tercera al script de instalación (instalación de Java) y la última al funcionamiento de ELK (reindexado). Todas ellas se han podido resolver correctamente y es por ello que se han podido completar algunos de los requisitos (4, 1 y 13 respectivamente) y validaciones que requerían el borrado de datos, para el caso del reindexado.

6. VALIDACIÓN

La verificación de la funcionalidad completa de cada elemento software que compone este proyecto se han realizado pruebas en distintos entornos: Preproducción y Producción.

6.1. Entorno de Preproducción

La etapa preproductiva se ha realizado durante y tras el desarrollo con el propósito de comprobar el funcionamiento de ELK, así como de los scripts de instalación y despliegue, con las versiones 14 y 16 de Ubuntu y Xubuntu.

Para poder permitir una mejor flexibilidad, todas estas pruebas se han realizado en entornos virtuales en "VirtualBox" de Oracle, lo cual permite replicar, modificar y borrar las instancias de cada máquina en caso de necesidad.

Estas máquinas están construidas sobre los archivos ISO originales de los sistemas Ubuntu y Xubuntu de versiones 14 y 16 (un total de 4 entornos) con las mismas especificaciones de RAM y de espacio de disco que la máquina de Producción, pero sin las limitaciones de la propia red del departamento (comentadas en el punto 6.2).

6.2. Entorno de Producción

Este entorno es una máquina de acceso remoto del departamento, en concreto es un Xubuntu con la versión 14. La diferencia con el entorno preproductivo es el software preinstalado en la máquina y las limitaciones en la red:

- Software preinstalado: lo único destacable es la versión 1.7 de Java que ya disponía el sistema (Elasticsearch requiere la versión 1.8).

- Limitaciones de la red: no se permite una instalación de Java 1.8 a partir del mandato "apt-get install"

Debido a estas diferencias se modificó el script de instalación, como se describe en el punto 4.4.

6.3. Pruebas realizadas.

Las pruebas son las mismas en ambos entornos y han ayudado a garantizar la funcionalidad de los objetivos y requisitos, a determinar las limitaciones existentes y a detallar los documentos realizados, especialmente el Manual de Usuario.

Lista de pruebas realizadas y resultados:

- **Pruebas con cada archivo de log por ELK sin análisis previos**: Validadas, la herramienta es capaz de recorrer todos los archivos log, indexar la información y mostrar las visualizaciones que corresponden a los datos introducidos.

- **Pruebas con cada archivo de log por ELK con análisis previos**: Validadas, la herramienta no analiza los archivos de log, por lo que no indexa información nueva.

- **Pruebas con cada archivo de log por ELK eliminando los índices de Elasticsearch**: Validadas, la herramienta no analiza los ficheros de logs, porque los punteros de Logstash apuntan al final de cada archivo.

- **Pruebas con cada archivo de log por ELK eliminando los datos guardados y los índices de Elasticsearch**: Validadas, la herramienta se comporta como en el primer caso, analiza los archivos de log e indexa la información.

- **Pruebas con cada archivo de log por ELK con logs erróneos**: Validadas, Logstash reporta o no analiza las líneas (siendo este su comportamiento ante estos casos).

- **Análisis de cada archivo de log por ELK con una mala configuración en los ficheros de Logstash (errores inducidos)**: Validado, Logstash reporta de una anomalía en la configuración y detiene el despliegue (comportamiento por defecto).

- **Pruebas con el límite de elementos visualizados en Kibana:** Probado con los logs que contienen conceptos. Con, aproximadamente, más de tres mil conceptos no se recomienda la visualización de tipo “TOP 100” o mayores.

- **Pruebas con las representaciones de datos en histogramas en Kibana**: Validadas, todas las visualizaciones solicitadas funcionan correctamente.

- **Pruebas de ejecución con los scripts de instalación y despliegue**: Validadas, los scripts funcionan correctamente y muestran la información necesaria.

- **Pruebas de ejecución con el script de despliegue con permisos administrativos**: Probado, Elasticsearch no permite el desplege con el usuario root (se debe arrancar sin permisos administrativos).

6.4. Otros recursos usados para la validación

En algunas situaciones se han utilizado recursos web o pruebas a una escala menor para comprobar el funcionamiento de los componentes. Algunos de ellos son:

- **RegExr** [13]: una página web que posee un motor para comprobar si la expresión regular usada es correcta, respecto a un texto que se puede proporcionar.

- **Grok Debugger** [14]: otra página web para comprobar expresiones de Grok de Logstash, aunque está limitada por el diccionario de patrones de Grok.

- **Pruebas locales con Ruby**: para poder comprobar la validez y funcionamiento de los scripts de Ruby que tiene la configuración de Logstash.

6.5. Resultados finales en Kibana

En éste último apartado se muestran los resultados de los desarrollos y pruebas realizadas, para demostrar la realización del cometido principal de este proyecto: la visualización de la información destacada de los logs.

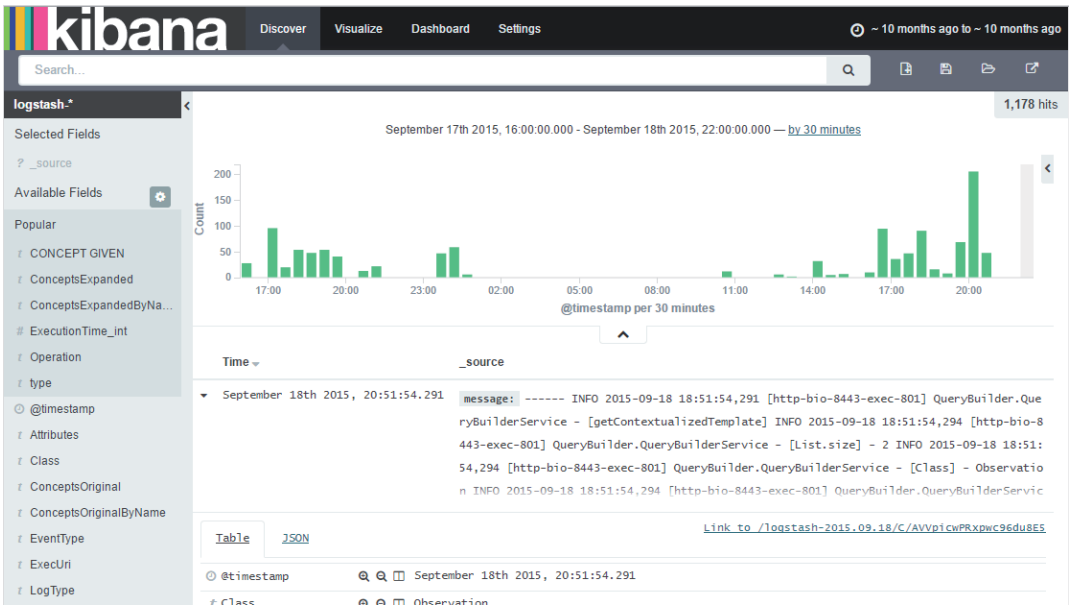

Imagen 3 – Interfaz Discover de Kibana

Como se puede observar en la "Imagen 3" anterior, se pueden obtener con detalle todos los logs analizados de los ficheros de logs del proyecto, destacándose con índices aquella información que aporta valor al usuario. Además Kibana permite hacer filtros y búsquedas sobre la información que se quiera destacar.

En este apartado, recomiendo la lectura del anexo "Documento de usuario" para más información sobre el contenido de Kibana.

En las siguientes ilustraciones se pueden observar algunas de las gráficas solicitadas para su visualización:

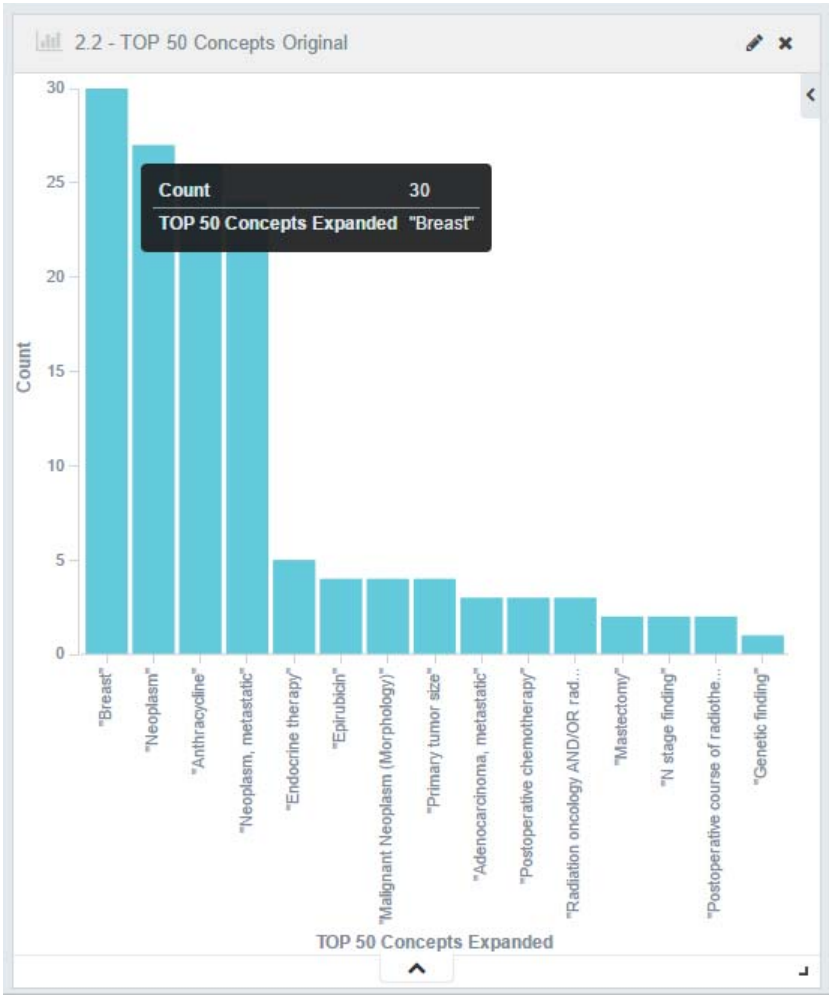

Imagen 4 – Top 50 conceptos de logs tipo Original

En esta "Imagen 4" se pueden observar la visualización de los nombres y descripciones de los conceptos comentados en el requisito 1.

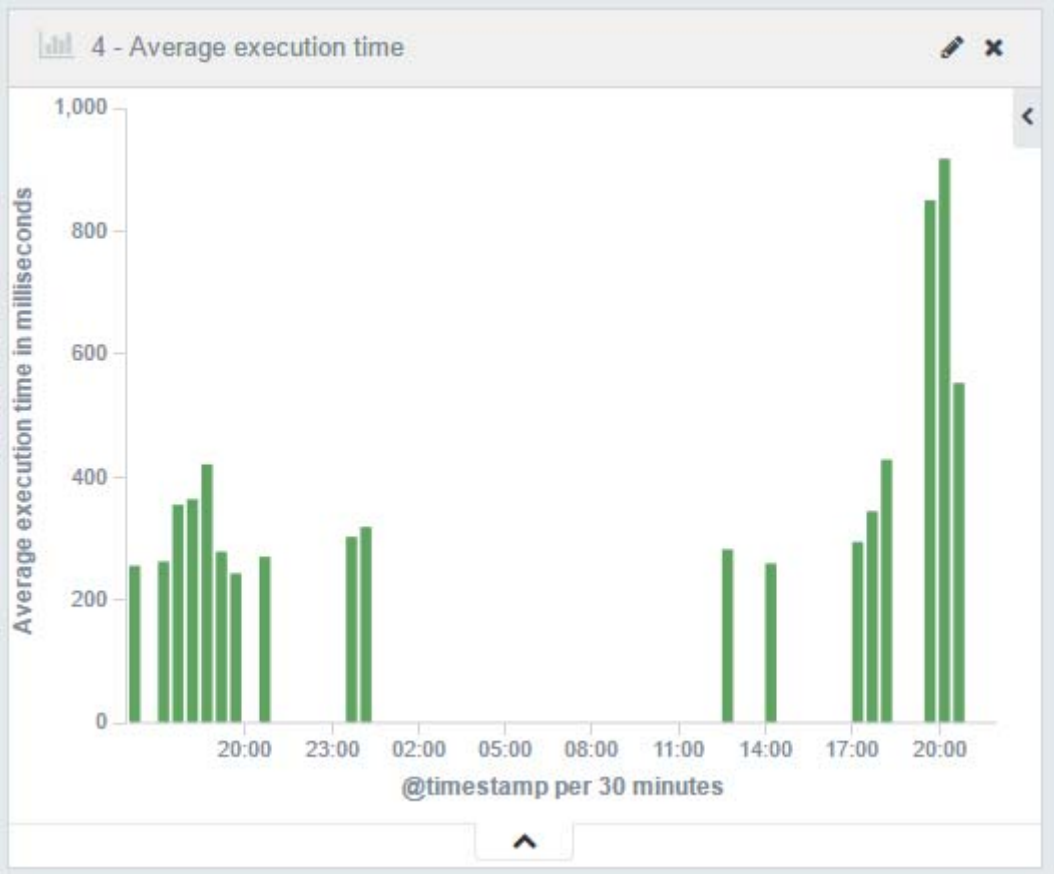

Imagen 5 – Media de tiempo de ejecución

Este tiempo medio de ejecución en un histograma (ver "Imagen 5") se solicita en el requisito 11.

Imagen 6 – Dashboard completo.

Por último, en esta "Imagen 6", sin demasiado detalle, se puede observar el Dashboard completo, con el conjunto de todas las visualizaciones solicitadas, entre las fechas que especifican los logs.

7. LÍNEA FUTURA

El proyecto permite posibilidades de mejora, no solo son dentro de su propio ámbito, sino también para la utilización de ELK para otros proyectos.

7.1. Mejoras internas del proyecto

El propio proyecto se puede perfeccionar en los siguientes apartados:

1. **Optimización de la configuración de Logstash**. Mediante la modificación de los nombres de los índices o la eliminación de todos aquellos índices que no se destaquen para su uso. Esta optimización aporta una mayor claridad al usuario.

2. **Incrementar la seguridad**. Por defecto Kibana no ofrece ningún mecanismo de control de acceso o de privilegios de usuario, por lo que se pueden tomar dos opciones:
   2.1. Control mediante proxy: ELK trabajará a nivel de localhost y solo se podrá acceder a sus herramientas mediante un proxy con la máquina.
   2.2. Con la herramienta de "Shield" ofrecida por Elastic: ofrece la seguridad comentada a Elasticsearch y a Kibana, pero no es gratuita.

3. **Mejora de la monitorización del sistema**. Kibana no tiene implementado ningún sistema de alertas o avisos, por lo que se puede:
   3.1. Crear dicho apartado con un backend que recoja la información del Elasticsearch o del sistema en general y envíe notificaciones si se produce algún evento inusual.
   3.2. Tener, con las herramientas de "Beats" y "Marvel" de Elastic, un mayor control de todo el sistema. Aunque el uso de Marvel no es gratuito.

7.2. Mejoras aplicables a otros proyectos

ELK está pensado para trabajar con grandes cantidades de logs de distinta procedencia, por lo que una posible mejora para otros proyectos que generen logs es la de aplicar este sistema con ellos y mantener una mejor monitorización de todos los proyectos en curso.

Incluso si la máquina donde se ha instalado el ELK no fuera suficiente por cuestiones de recursos, tanto Logstash como Elasticsearch pueden trabajar con distintos nodos, de hecho, el ELK optimiza su rendimiento cuando se escalan horizontalmente sus procesos.

8. CONCLUSIONES

Sobre el ELK:

Se observa que, sin conocimientos previos de la herramienta, la curva de aprendizaje es bastante alta. Sobre todo para entender el funcionamiento y la relación entre las herramientas, y también en la configuración del Logstash, por tener una gran cantidad de plugins para investigar.

En concreto, la configuración del Logstash es lo que ha llevado más tiempo de asimilación para aumentar la efectividad en el desarrollo, puesto que hay muchas posibilidades de configurarlo y no todas ellas permiten generar los resultados deseados. Personalmente, he encontrado situaciones en las que he tenido que infórmame mejor vía web (en la página de soporte de Elastic o en foros) para poder cumplir con las expectativas del proyecto y omitir las funcionalidades erróneas o que no cubrían el requisito en su totalidad.

Así mismo, sobre el ELK se han observado algunas limitaciones funcionales, registradas en el anexo: Documento de Usuario (en el punto 5: “Limitaciones de ELK).

De las cuales se destacan:

- El gran uso de memoria que hacen las tres herramientas: Logstash durante el análisis de logs, Kibana en las visualizaciones de gran cantidad de parámetros (en el navegador) y en el Elasticsearch.

- La limitación en el debug de la configuración de Logasth (en la mayoría de los casos no se aporta suficiente información).

- La falta de seguridad (control de accesos) que tiene Kibana.

Aportaciones recibidas durante el proyecto:

- Conocimientos adquiridos:
  - Configuración y uso del ELK.
  - Regrex.
  - Ruby.

- Mejora progresiva en la comunicación y especificación de requisitos.

- Repasado de nociones de scripts de BASH.

- Mejora en el redactado y en la exposición de elementos clave en las documentaciones.

Conclusión final sobre el proyecto:

Finalmente, la impresión sobre todo lo realizado es buena, ya que lo demandado durante la realización del proyecto se ha resuelto y los objetivos han sido cumplidos, como se ha podido observar en los apartados anteriores, a pesar de las dificultades encontradas.

9. REFERENCIAS

[1] Elastic. 2016. An Introduction to the ELK Stack Elastic [En línea]. Disponible: https://www.elastic.co/webinars/introduction-elk-stack

[2] Elastic. 2016. The Elastic Stack [En línea]. Disponible: https://www.elastic.co/products

[3] Vanderzyden, John. 1 de septiembre del 2015. What is Elasticsearch, and How Can I Use It? [En línea]. Disponible: https://qbox.io/blog/what-is-elasticsearch

[4] Wikipedia. 7 de junio del 2016. Elasticsearch [En línea]. Disponible: https://es.wikipedia.org/wiki/Elasticsearch

[5] Elastic. 2016. Logstash Introduction [En línea]. Disponible: https://www.elastic.co/guide/en/logstash/current/introduction.html

[6] Elastic. 2016. Introduction (Kibana) [En línea]. Disponible: https://www.elastic.co/guide/en/kibana/current/introduction.html

[7] Ministerio de Sanidad, Servicios Sociales e Igualdad. ¿Qué es Snomed-CT? [En línea]. Disponible: http://www.msssi.gob.es/profesionales/hcdsns/areaRecursosSem/snomed-ct/quees.htm

[8] Stackoverflow. 6 de agosto del 2014. Logstash date parsing as timestamp using the date filter [En línea]. Disponible: http://stackoverflow.com/questions/25156517/logstash-date-parsing-as-timestamp-using-the-date-filter

[9] Elastic.1 de septiembre del 2015. How to set @timestamp timezone? [En línea]. Disponible: https://discuss.elastic.co/t/how-to-set-timestamp-timezone/28401

[10] Stackoverflow. 13 de agosto del 2009. Ruby: Search file text for a pattern and replace it with a given value? [En línea]. Disponible: http://stackoverflow.com/questions/1274605/ruby-search-file-text-for-a-pattern-and-replace-it-with-a-given-value

[11] Askubuntu. 7 de agosto del 2011. How can I install Sun/Oracle's proprietary Java JDK 6/7/8 or JRE? [En línea]. Disponible: http://askubuntu.com/questions/56104/how-can-i-install-sun-oracles-proprietary-java-jdk-6-7-8-or-jre

[12] Stackoverflow. 23 de octubre del 2013. How to force Logstash to reparse a file? [En línea]. Disponible: http://stackoverflow.com/questions/19546900/how-to-force-logstash-to-reparse-a-file

[13] Skinner, Grant. 4 de mayo del 2016. RegExr v2.1 [En línea]. Disponible: http://regexr.com/

[14] Ethier, Nick. 26 de diciembre del 2015. Grok Debugger [En línea]. Disponible: https://grokdebug.herokuapp.com/

10. ANEXOS

Todos los anexos se encuentran incluidos con la entrega de esta memoria.

10.1. Fichero de configuración de Logstash

Es el fichero “ConfigBase.conf”.

10.2. Fichero de patrones

Es el fichero “patterns”.

10.3. Script de instalación

Es el fichero “Script_ELK_Installer.sh”.

10.4. Script de despliegue

Es el fichero “Script_ELK_Starter.sh”.

10.5. Documento de instalación

Es el documento denominado “Manual de Instalación de ELK V3.0.pdf”.

10.6. Documento de usuario

Es el documento denominado “Manual de Usuario de ELK V2.0.pdf”.

Este documento esta firmado por

Firmante	CN=tfgm.fi.upm.es, OU=CCFI, O=Facultad de Informatica - UPM, C=ES
Fecha/Hora	Mon Jul 04 23:26:30 CEST 2016
Emisor del Certificado	EMAILADDRESS=camanager@fi.upm.es, CN=CA Facultad de Informatica, O=Facultad de Informatica - UPM, C=ES
Numero de Serie	630
Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)