GRADO EN TECNOLOGÍAS DE LA TELECOMUNICACIÓN

Curso Académico 2018/2019

Trabajo Fin de Grado/Máster

REPRESENTACIÓN DE DATOS DE GRIMOIRE LAB CON GRAFANA

Autor : Jonathan Cano Picazo

Tutor : Dr. Jesús María González Barahona

Trabajo Fin de Grado/Máster

Representación de Datos de Grimoire Lab con Grafana

Autor : Jonathan Cano Picazo

Tutor : Dr. Jesús González Barahona

La defensa del presente Proyecto Fin de Carrera se realizó el día de de 20XX, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 20XX

Dedicado a

mi familia / mis amigos / mis abuelos,

pero sobre todo a mis padres, que me apoyaron desde el principio.

II

Agradecimientos

Todo comenzó en el momento en el que recibí mi primer ordenador personal. Desde aquel instante la tecnología entró por mis ojos, y tuve bastante claro que quería dedicar mi vida entorno a ella. Han pasado ya unos 18 años desde entonces, y no me arrepiento de mi decisión.
Ha sido una travesía llevadera en algunos momentos, pero muy dura en otros tantos. Ha habido momentos de mucha diversión, desde el momento en el que conocí a mis primeros compañeros de clase, los cuales siguen siendo íntimos amigos. Pero también he pasado momentos difíciles, con esos temidos exámenes finales, y por desgracia bastantes recuperaciones...
Pero finalmente logré mi objetivo. Y esto no hubiese sido posible de no ser por el apoyo incondicional de mi familia: mis padres, Mª Ángeles y Francisco, y mis hermanos, Rubén e Ismael. No tengo palabras para agradecer toda la ayuda que me habéis proporcionado durante todos estos años. Aunque también recibí mucho apoyo por parte de mi abuelo, que deseaba que su nieto se labrase un futuro como ingeniero, y así sentirse completamente orgulloso al hablar de él y de sus éxitos. Y por suerte, va a vivir para contarlo...
Y por último, y no menos importante, querría darle las gracias también a mi gran amigo Javier Fernández Morata, quién ha sido quizás mi gran apoyo durante estos últimos y duros años de la carrera.

Resumen

Aquí viene un resumen del proyecto. Ha de constar de tres o cuatro párrafos, donde se presente de manera clara y concisa de qué va el proyecto. Han de quedar respondidas las siguientes preguntas:

- ¿De qué va este proyecto? ¿Cuál es su objetivo principal?
- ¿Cómo se ha realizado? ¿Qué tecnologías están involucradas?
- ¿En qué contexto se ha realizado el proyecto? ¿Es un proyecto dentro de un marco general?

Lo mejor es escribir el resumen al final.

Summary

Here comes a translation of the “Resumen” into English. Please, double check it for correct grammar and spelling. As it is the translation of the “Resumen”, which is supposed to be written at the end, this as well should be filled out just before submitting.

Índice general

Capítulo 1

Introducción

1.1. Contexto

En la actualidad, gracias al auge de las tecnologías de la información y comunicación, un usuario es capaz de recibir continuos datos desde múltiples fuentes. Uno de los grandes usos de esta tecnología es en el ámbito de la publicidad. Los anuncios se pueden encontrar de múltiples formas: vallas publicitarias, vehículos públicos, pantallas LED... Es algo que impacta de manera muy directa en el consumidor, y puede decantar la decisión de, por ejemplo, comprar un producto o no.

Este pequeño ejemplo permite introducir un nuevo término, el cual está íntimamente relacionado. Hablamos de la visualización o representación de datos. Es una herramienta, que utilizada de la forma adecuada, permite al usuario explorar de un vistazo la parte de información que pueda interesarle más o menos. Al igual que en el caso de la publicidad, existen diversas maneras de mostrar dicha información: utilizando un aspecto más visual o llamativo, mediante gráficos o cuadros de mando; o algo más enfocado a observar un gran número de registros, datos estadísticos... Todo ello nos permite observar la correlación existente entre los diferentes datos mostrados.

La manera de representar los datos, y su aspecto visual final, son determinantes a la hora de presentarlo al usuario. Para ello existen múltiples herramientas que nos facilitan el uso y manejo de cada bit recolectado. Y esto es precisamente lo que se aborda en este proyecto. Explorar esas herramientas y explotar al máximo sus posibilidades.

Por ese mismo motivo, las grandes empresas dedican muchos recursos a este sector, ya que es una manera muy eficaz de, por ejemplo:

- Observar la escala evolutiva de un proyecto.
- Tener un seguimiento en tiempo real de diferentes infraestructuras (webs, BBDD, radio-enlaces...).
- Diagnosticar problemas (de rendimiento, uso, congestión...).
- Y por supuesto, atraer la atención de futuros clientes/patrocinadores.

Esta fue la visión general que mi tutor, el Dr. Barahona, me dio sobre el proyecto en el que me acabaría embarcando. No tardé mucho en decantarme por la idea ofertada, ya que me entró mucho por los ojos, me impactó. Tuvo un efecto similar al de un consumidor con un anuncio, como se ha comentado al principio.

1.2. Estructura de la memoria

Debido al gran volumen de datos del proyecto en el que nos encontramos, se ha de seguir una determinada organización, para proporcionar una correcta lectura y comprensión. Esta tesis, por lo tanto, se estructura de la siguiente manera:

- **Capítulo 1: Introducción**. Se presenta la idea inicial del proyecto y como surgió. Además, se proporciona una pequeña estructura de la tesis, explicando el contenido de cada sección.
- **Capítulo 2: Objetivos**. En este apartado se define más concretamente la finalidad de este proyecto y cuales son sus requisitos u objetivos finales.
- **Capítulo 3: Estado del arte**. Aquí nos encontraremos con una descripción general de todas las herramientas utilizadas en este proyecto (historia, características, uso...).
- **Capítulo 4: Diseño e implementación**. Se presentan y desarrollan las diferentes etapas que han llevado al proyecto hasta su estado final. Estas etapas están basadas en el algoritmo Scrum, para desarrollo ágil de software.

- **Capítulo 5: Resultados**. Análisis de los resultados finales obtenidos.
- **Capítulo 6: Conclusiones**. En esta sección se contrastarán los datos obtenidos con los objetivos iniciales del proyecto, para comprobar si se han cumplido las expectativas. Además, se proponen mejoras y posibles proyecciones futuras, así como una valoración general del proyecto y de los conocimientos aplicado y aprendidos.

Capítulo 2

Objetivos

2.1. Objetivo general

En este proyecto se compone de 4 objetivos principales:

- Representación de datos con Grafana[1] gracias al uso de Grimoire Lab[2], la herramienta que nos permite la recolección y almacenamiento de datos extraídos de Git[3] y Github[4].

- Unificación de ambas en un mismo *docker* o contenedor gracias a Docker Hub, automatizando así el despliegue de ambas aplicaciones en una misma imagen.

- Despliegue web de este software unificado, permitiendo un acceso más rápido e intuitivo al usuario.

- Comparación de las dos herramientas de representación utilizadas, Grafana y Kibana[5]. Para ello se exploran los límites de cada una de ellas, ahondando sobre todo en el apartado de plugins que ofrece la primera.

[1] `http://www.grafana.com`
[2] `https://chaoss.github.io/grimoirelab/`
[3] `https://git-scm.com/`
[4] `https://https://github.com/`
[5] `https://www.elastic.co/es/products/kibana`

2.2. Objetivos secundarios

Para conseguir los resultados propuestos en el objetivo principal, ha sido necesario también manejar al completo distintas herramientas, como es el caso de Elasticsearch[6], Kibana y Docker Hub[7].

Primeramente, se abordó el tema de las bases de datos no relacionales. En este proyecto, la mayor parte de la información recolectada se almacena en Elasticsearch. Por lo tanto, ha sido necesaria la formación en este tipo de BBDD, ya que siempre había trabajado con relacionales. Además, enlaza directamente con la segunda herramienta, Kibana. Esta última es un complemento de visualización de datos para Elasticsearch, y es el software utilizado por defecto por Grimoire Lab. Una vez aprendí su total manejo, ha servido de guía para obtener resultados similares a los proporcionados por Grafana, y realizar así su posterior comparación.

2.3. Planificación temporal

La realización de este proyecto se ha llevado a cabo durante un periodo de 12 meses. La mayor parte del tiempo coincidió durante mi último curso académico, mientras que los últimos meses se solapó con mi trabajo actual. En este lapso, el proyecto ha seguido diferentes fases:

- Elección del proyecto. En primer lugar, me reuní con mi tutor del proyecto, Jesús, para que me ayudase a elegir un proyecto que se adaptase a lo que estaba buscando. Me presentó dos posibles ideas, quedándome finalmente con la presente, ya que me llamó mucho más la atención.

- Aprendizaje de las tecnologías a utilizar. Esta fue la parte más tediosa y larga del proyecto. Hizo falta leer mucha documentación e instalar bastantes dependencias, para adecuar el entorno sobre el que desarrollar la idea que Jesús y yo teníamos en mente.

- Implementación de los objetivos propuestos.

- Redacción de la memoria. Esta fase se fue solapando con la anterior, rellenando el documento presente ha medida que se terminaban objetivos concretos del proyecto.

[6] `https://www.elastic.co/es/products/elasticsearch`
[7] `https://hub.docker.com/`

Capítulo 3

Estado del arte

Como se ha comentado en el capítulo anterior, existen infinidad de herramientas para representar visualmente la información. Estas dependen a su vez de muchas otras, que conectadas, nos permiten reproducir un resultado final muy llamativo. En este capítulo, por tanto, se abordarán todos los aspectos relacionados con el software utilizado para la elaboración de este proyecto. Para poder presentar todas las tecnologías, debemos empezar por lo básico, lo primero que se realiza, la obtención de la información. Para ello nos ayudamos de dos plataformas: Git y Github.

3.1. Git

Git es un software de control de versiones (VCS, *Version Control System*) originalmente diseñado por Linus Torvalds en 2005. Su propósito es llevar registro de los cambios en archivos, permitir recuperar versiones anteriores de estos o actualizar a versiones posteriores. Está pensado para coordinar el trabajo que varias personas realizan sobre archivos compartidos. Actualmente, Git es el VCS más usado en el mundo, por encima de sus competidores directos, como pueden ser CVS[1], SVN[2] o Mercurial[3]. Su funcionamiento es muy sencillo. Partimos de un espacio de trabajo personal o *repositorio*. A partir de aquí, procedemos a crear y/o modificar los archivos de nuestro proyecto. Para poder guardar el estado en el que se encuentra en ese momento nuestro repositorio, primero tenemos que indicarle a Git que archivos queremos que

[1] `https://www.gnu.org/software/trans-coord/manual/cvs/cvs.html`
[2] `https://subversion.apache.org/`
[3] `https://www.mercurial-scm.org/`

preserve a lo largo del tiempo. Una vez añadidos, basta con "comprometer" los cambios (hacer un *commit*) para que Git haga una captura o *snapshoot* del estado del repo. A partir de aquí, podemos crear nuevas ramas de trabajo paralelas a una versión dada, volver a versión más antigua del proyecto, borrar versiones...

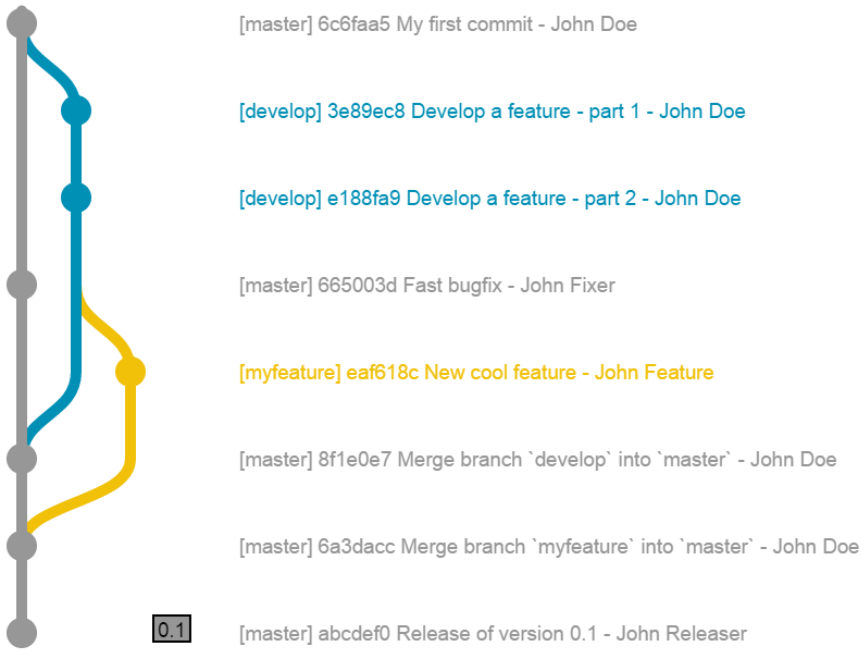

Figura 3.1: Arquitectura en árbol de Git

Este tipo de estructura es la que ha hecho a Git tan famoso hoy en día, y ha permitido tener grandes grupos de personas trabajando remota y paralelamente en un mismo proyecto. Para poder trabajar de manera remota y no local, surgió la herramienta que se presenta a continuación, Github.

3.2. Github

GitHub es un sitio web que permite alojar proyectos utilizando el sistema de control de versiones Git. Fue fundado en 2008, y se usa principalmente para proyectos de desarrollo de código fuente conjuntos. El software que opera GitHub fue escrito en Ruby on Rails. Actualmente, es una de las webs líder en su sector, muy por delante de sus competidoras directas,

como son Buddy[4] o Bitbucket[5]. La principal diferencia entre una herramienta y otra, reside en que Git solo funciona mediante un terminal o línea de comandos, mientras que Github proporciona una interfaz gráfica muy vistosa. Esta tiene multitud de opciones, entre las que destacan, entre muchas otras:

- Cada usuario tiene su perfil único, donde se muestra en que proyectos está trabajando, todos sus contribuciones pasadas, etc.

- Posibilidad de contribuir en proyectos públicos, haciendo *fork* de estos (copia editable del diseño de otro usuario), y si se permite, un posterior *pull request* (acción de validar un código que se va a hacer *merge* de una rama a otra).

- Páginas web dedicadas a cada proyecto, contando con *wiki*, documentación, registro de *commits* anteriores, *issues*, etc.

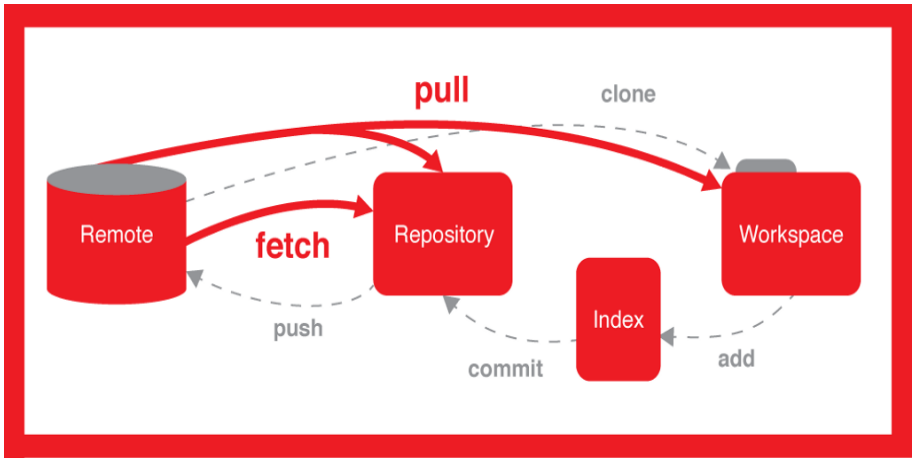

Figura 3.2: Funcionamiento básico de Github

[4] https://buddy.works/
[5] https://bitbucket.org/

3.3. Elasticsearch

Elasticsearch es un servidor de búsqueda y análisis basado en Lucene[6]. Es de código abierto, basado en Java[7], y con licencia Apache[8]. Trabaja conjuntamente con un motor de recopilación de datos llamado Logtash[9], y una plataforma de análisis y visualización llamada Kibana[10]. Juntos, forman el proyecto ELK (acrónimo de los nombres de las tres herramientas). Entre sus principales características, se puede destacar que:

- Posee una intuitiva API con interfaz web RESTful.
- Provee un motor de búsqueda de texto completo, distribuido y con capacidad de multi-tenencia con documentos JSON[11].
- Es un sistema de base de datos no relacional, por lo que tiene una gran velocidad de respuesta al trabajar con conjuntos muy grandes de datos, debido al uso de índices invertidos distribuidos.
- Viene completamente integrado con Logstash y Kibana, por lo que facilita el uso de las tres herramientas a la vez (recolección/tratado, guardado y visualización).

3.4. Kibana

Kibana es un *plugin* o complemento de Elasticsearch, de código abierto. Permite la visualización de datos previamente indexados por la herramienta anterior. Ofrece múltiples posibilidades para crear diferentes representaciones: gráficos de barras horizontales/verticales, lineales, *scatter*/dispersión, de tarta, circulares, etc. Además, permite diseñar tablas, listas, tops, medidores, mapas de calor... Al igual que Elasticsearch, Kibana es fácilmente accesible desde cualquier navegador de manera local. A partir de ahí, se pueden crear paneles desde cero, o utilizar alguna de las bibliotecas de la herramienta para importar paneles predeterminados, y hacerse una idea de un posible resultado final.

[6] `https://lucene.apache.org/`
[7] `https://www.java.com/es/`
[8] `http://httpd.apache.org/`
[9] `https://www.elastic.co/es/products/logstash`
[10] `https://www.elastic.co/es/products/kibana`
[11] `https://www.json.org/`

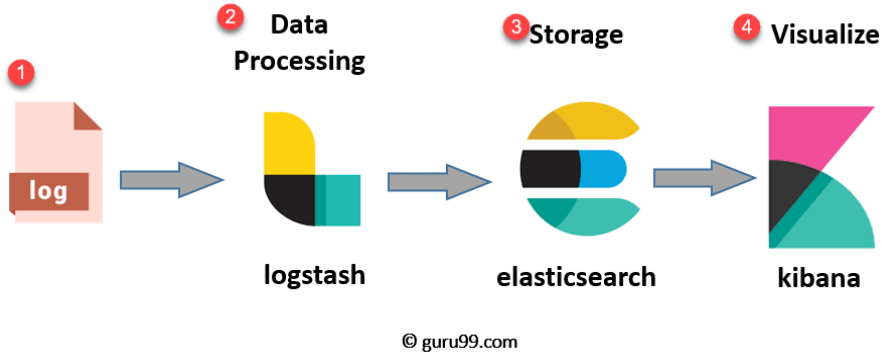

Figura 3.3: Esquema de funcionamiento de *ELK Stack*

3.5. Grafana

De manera similar a la herramienta anterior tenemos a Grafana, otro potente software de código abierto basado en licencia de Apache, escrito en lenguaje *Go*[12]. Permite también la visualización de información almacenada en múltiples bases de datos. Trabaja especialmente bien con BBDD de series de tiempo, tales como Graphite, InfluxDB... Originalmente comenzó como un componente de Kibana, aunque en la actualidad se considera como proyecto independiente. Entre las características principales, se puede destacar:

- Es una herramienta RESTful. Se puede ejecutar desde cualquier terminal, y a partir de ahí es accesible desde cualquier navegador, al ser una API HTTP.
- Es multiplataforma, y no tiene ninguna dependencia con otros programas, al contrario que Kibana.
- Además de los cuadros de mandos básicos (gráficos, tablas, *heatmaps*...) permite añadir muchos más mediante *plugins* de fácil instalación.
- Tiene una gran comunidad detrás, la cual amplía drásticamente las maneras de representar los datos, mediante los citados complementos.
- Compatibilidad con una de bases de datos: Elasticsearch, MariaDB[13], MySQL[14], Post-

[12] `https://golang.org/`
[13] `https://mariadb.org/`
[14] `https://www.mysql.com/`

greSQL[15], CloudWatch[16]...

- Gestión muy intuitiva de usuarios y *dashboards*[17]. Esto permite que cada persona posea unos cuadros de mandos propios, y que pueda realizar *snapshoots* de estos, para poder compartirlos públicamente.

- PONER ALGUNA MÁS

IMAGEN DE DASHBOARD PROPIO

3.6. GrimoireLab

GrimoireLab es un conjunto de herramientas de código libre, orientada a la investigación analítica de datos. Ha sido desarrollada por la empresa española Bitergia[18]. Permite extraer datos desde muchas fuentes distintas, y producir análisis y visualizaciones de estos. Su uso ha sido crucial para el propósito de esta memoria, pues gracias a su herramienta Perceval[19], nos permite recolectar información de más de 20 fuentes diferentes, pertenecientes a repositorios de Git y a proyectos de Github. Estos proyectos abarcan desde *issue trackers* (rastreadores de problemas) como Jira[20] o Bugzilla[21], *mailing lists* o listas de correo, diferentes fuentes de Stackoverflow[22], etc. Actualmente, GrimoireLab forma parte del proyecto CHAOSS, perteneciendo también a la Fundación Linux.

3.7. Docker Hub

Docker Hub es un conjunto de repositorios que permite alojar contenedores de imágenes de distintos tipos: proyectos de desarrollo de la comunidad, proyectos de código abierto o vendedores independientes de código (ISV), que construyen y distribuyen su código en contenedores.

[15] https://www.postgresql.org/
[16] https://aws.amazon.com/es/cloudwatch/
[17] https://www.40defiebre.com/que-es/dashboard
[18] https://bitergia.com/
[19] https://chaoss.github.io/grimoirelab-tutorial/perceval/intro.html
[20] https://www.atlassian.com/es/software/jira
[21] https://www.bugzilla.org/
[22] https://stackoverflow.com/

Los usuarios pueden tener acceso a repositorios públicos gratuitos para almacenar y compartir sus propias imágenes, u optar por un plan de pago, permitiendo tener repositorios privados. La empresa Docker fue creada en creada en 2013 por Solomon Hykes. Está basada en el lenguaje de programación Go, y cuenta con licencia Apache. La idea inicial del proyecto fue poder crear un entorno virtual que permita empacar todas las dependencias, herramientas y código necesario para que un número determinado de aplicaciones se ejecute de manera rápida y segura. Este paquete ejecutable, en esencia, una imagen de contenedor. Entre sus características principales, destacan:

- Tiene la mayor comunidad de contenedores de imágenes del mundo. La mayoría son oficiales y gratuitas, por lo que podemos hacer *pull* de ellas directamente (ejemplos serían Elasticsearch, Grafana, GrimoireLab, Ubuntu, MongoDB, etc.).

- Permite a los desarrolladores resolver el problema "works on my machine". Al tener entornos virtuales aislados, permite trabajar de manera simultánea en diferentes espacios específicos a cada situación.

- Tolera la construcción de imágenes de manera automática, desde contenedores alojados en Github o Bitbucket. Si enlazamos ambas plataformas, al hacer *push* se resuben las imágenes en Docker Hub.

AWS, JEKYLL, OVERLEAF

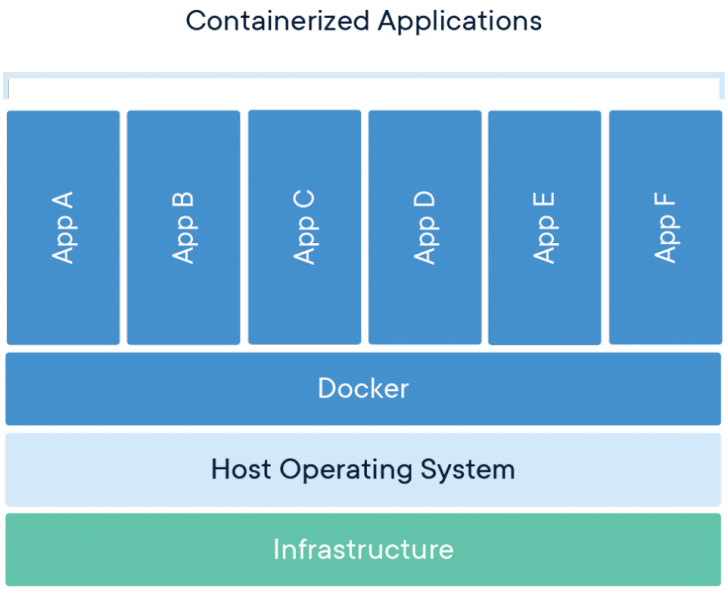

Figura 3.4: Esquema de funcionamiento de DockerHub

Capítulo 4

Diseño e implementación

En este capítulo abordaremos diferentes aspectos del proyecto, como son su estructuración, su desarrollo y su implementación. Primeramente, debemos mencionar la metodología utilizada en este proyecto, la cual es una modificación más simple del conocido **método SCRUM**[1].

4.1. SCRUM

Scrum es un metodología de trabajo enfocada al **desarrollo ágil de software**. Se basa en la aplicación de un conjunto de buenas prácticas de manera regular, para trabajar en equipo, y obtener el mejor resultado posible de proyectos. Su principal característica consiste en solapar las diferentes fases del desarrollo, en lugar de realizar una tras otra en un ciclo secuencial o en cascada. Para ello, el trabajo se subdivide en tareas más pequeñas, para así poder abordarlas concurrentemente. Estas subdivisiones se denominan *sprints*.

El ***sprint*** es el período de tiempo en el cual se lleva a cabo el trabajo en sí. En este proyecto su duración inicial era de una semana, aunque se fue ampliando debido al volumen de objetivos a realizar en cada *sprint*, los llamados ***sprint backlogs***.

Debido a que nuestro equipo de desarrollo es muy pequeño (dos personas), nos vimos obligados a modificar la metodología, a simplificarla. Por tanto, no estarán presentes aspectos característicos de Scrum, como pueden ser reuniones diarias, duración mínima de sprints (dos semanas), flexibilidad a cambios (no tenemos cliente), predicciones de tiempos, etc.

[1] `https://www.scrum.org/resources/blog/que-es-scrum`

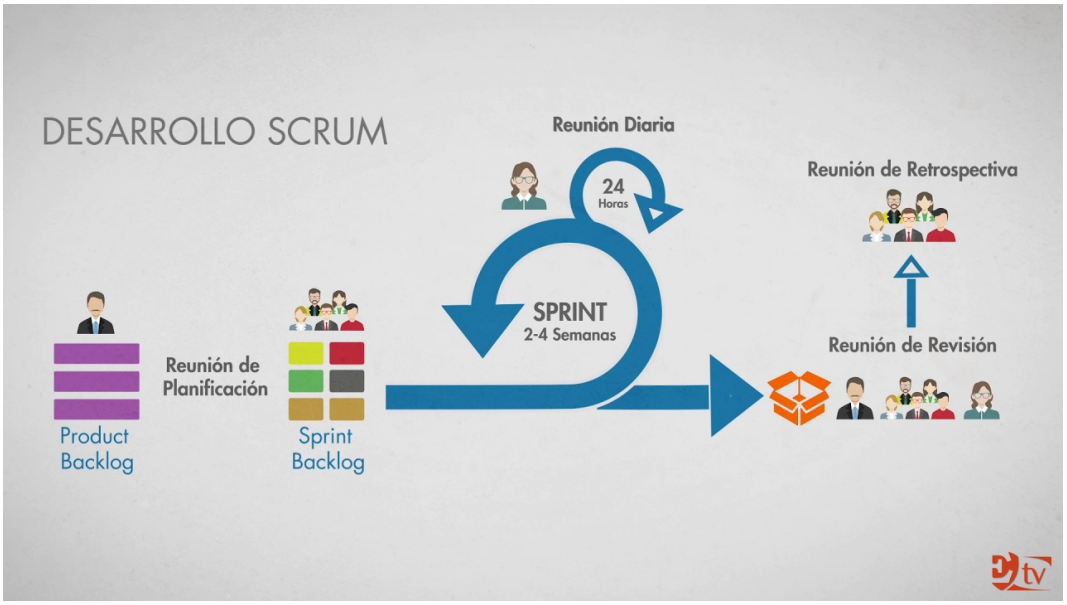

Figura 4.1: Esquema del entorno de trabajo SCRUM

4.2. Arquitectura general

A continuación, se presenta la estructura general del proyecto. Tal y como se ha comentado anteriormente, está dividida en los diferentes *scrums* realizados. En la Figura 4.2 se puede observar de un vistazo la totalidad del proyecto (SE PUEDE MEJORAR ESQUEMA??). Se estructura en cuatro partes:

- Bloques azules: DockerHub, GrimoireLab y Kibana.
- Bloques verdes: Grafana y plugins.
- Bloques morados: Creación de un contenedor de imagen que incluya todo.
- Bloque naranja: Comparación de las dos herramientas principales: Grafana y Kibana.

4.3. Sprint 1 - Docker Hub y GrimoireLab

Tal y como se ha mostrado en el esquema anterior, el primer paso del proyecto fue instalar los componentes y dependencias necesarias para poder visualizar nuestro primer dashboard. Para ello, utilizamos Docker Hub, herramienta mencionada en la sección 3.7, que nos permitirá utilizar un contenedor de imagen ya preparado para poder visualizar rápidamente diferentes

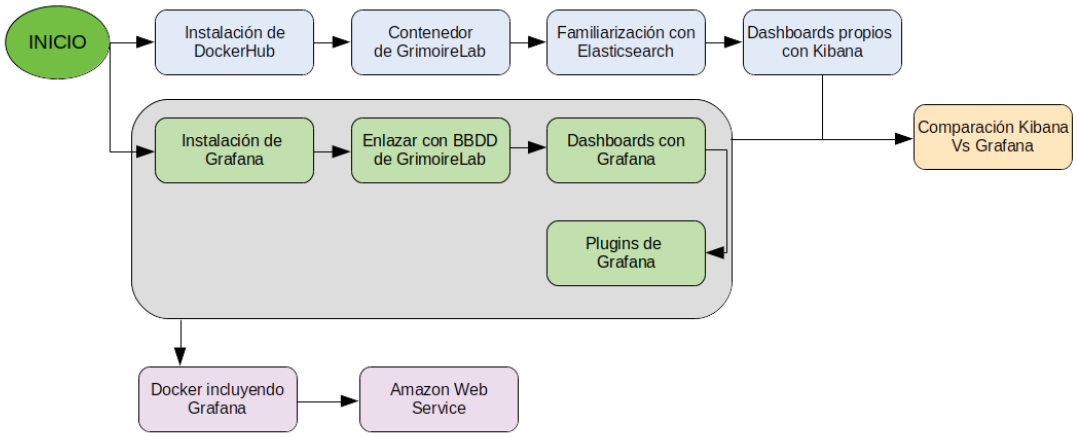

Figura 4.2: Diagrama de bloques del proyecto

paneles de control. Más adelante, en la sección 4.7.1, se profundiza en el funcionamiento de la herramienta.

4.3.1. Preparando el contenedor

El contenedor de imagen a utilizar es `grimoirelab/full`. Fue creado por el equipo CHAOSS, y contiene todos los elementos necesarios para producir dashboard completo y funcional: Elasticsearch, MariaDB, and Kibiter. El contenedor produce un dashboard del proyecto GrimoireLab, que nos servirá de guía para nuestros futuros paneles. Lo primero, por tanto, será la instalación de Docker Hub[2]. También es recomendable seguir los pasos de la post-instalación[3], para evitar tener que dar privilegios de acceso/administrador (usuario *root*), escribiendo 'sudo' al principio de cada sentencia).

Una vez instalada la herramienta siguiendo el manual (mediante repositorio o paquete de instalación), se procede a descargar el contenedor de imagen mencionado anteriormente. Basta con introducir la siguiente sentencia:

```
1 docker pull grimoirelab/full
```

[2] `https://docs.docker.com/install/linux/docker-ce/ubuntu/`
[3] `https://docs.docker.com/install/linux/linux-postinstall/`

Antes de poder lanzar el contenedor debemos preparar un archivo de configuración, para que este pueda recolectar adecuadamente los datos necesarios. Para ello, debemos crear el fichero `credentials.cfg` en una ruta de fácil acceso (/home/USER en mi caso). El fichero debería ser un *token*[4] de la API de Github, siguiendo el siguiente formato:

```
[github]
api-token = XXX
```

Ahora sí, se procede a lanzar el contenedor de imagen:

```
docker run -p 127.0.0.1:5601:5601\
-v $(pwd)/credentials.cfg:/override.cfg\
-t grimoirelab/full
```

Ahora, para poder visualizar nuestro primer dashboard, nos dirigimos a cualquier navegador e introducimos: `http://localhost:5601`. Se nos mostrará algo similar a esto: (IMAGEN KIBITER OVERWIEW). Todo esto disponible en el tutorial de la página del proyecto[5].

4.3.2. Funcionamiento de GrimoireLab

Antes de continuar explorando la herramienta de GrimoireLab, se decidió estudiar exhaustivamente su funcionamiento. Para ello, el esquema de la Figura 4.3 fue de gran ayuda:

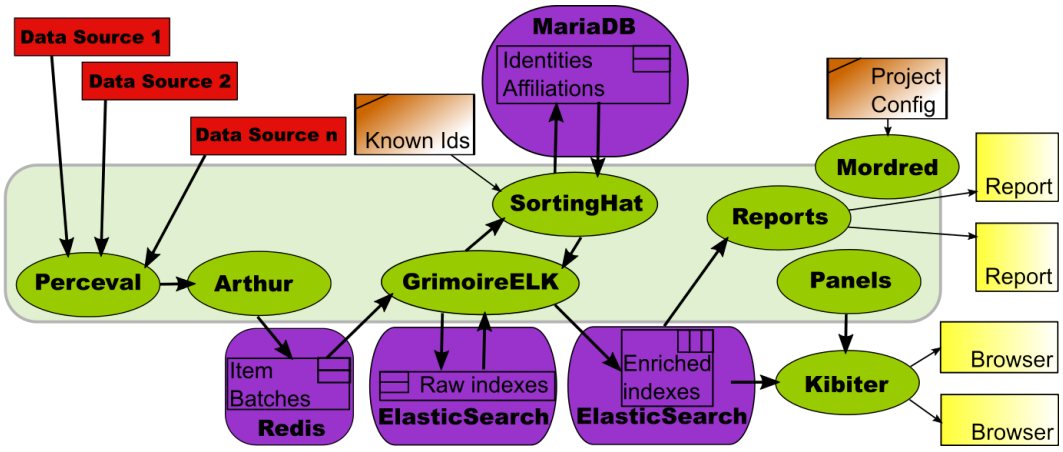

Figura 4.3: Esquema de funcionamiento de GrimoireLab

En la Figura se pueden apreciar todos los componentes de la herramienta, representados dentro del rectángulo verde. Las flechas sombreadas corresponden al flujo principal de los da-

[4] `https://github.com/settings/tokens`

[5] `https://chaoss.github.io/grimoirelab-tutorial/basics/dockerhub.html`

tos: comenzamos con Perceval (que extrae la información de las fuentes), siguiendo por Arthur (que programa lotes de recuperación y almacena resultados en Redis[6]), llegando a GrimoireELK (que almacena los datos extraídos como índices crudos, y los utiliza para producir índices enriquecidos, ambos en ElasticSearch), para finalmente llegar a Kibiter (herramienta "soft fork" de Kibana, para visualizar dashboards interactuables).

GrimoireELK (análogo al proyecto ELK visto en la Sección 3.3) utiliza SortingHat para almacenar todas las identidades que encuentra en una DB de MariaDB. SortingHat utiliza listas de identificadores conocidos (generalmente mantenidos en archivos de configuración) y heurísticas para fusionar identidades correspondientes a la misma persona e información relacionada (como afiliación).

Todo el proceso es configurado y orquestado por Mordred, el cual usa su propia configuración sobre, por ejemplo, qué fuentes de datos utilizar.

4.3.3. Exploración de dashboards con Kibiter/Kibana

Una vez aprendido el funcionamiento de GrimoireLab, comenzamos a explorar la cantidad de posibilidades que ofrece una de sus herramientas, Kibiter (que es un "soft fork" de Kibana, compartiendo todas las funciones principales de esta).

Siguiendo los pasos de la sección 4.3.1, tendremos inicializado el contenedor, el cual irá mostrando distintas líneas en el terminal a modo de *logs*, como se aprecia en la Figura 4.4. Una vez recolectada la información y enriquecidos los índices (puede demorar un tiempo), podremos acceder a la página en el navegador (http://localhost:5601), mostrándose así nuestro primer dashboard.

A continuación, si nos situamos en la barra superior de la página, podemos elegir entre las diferentes categorías de paneles a visualizar (todos pertenecientes al proyecto CHAOSS). Un ejemplo sería el de *Overview*, el cual mostrará algo similar a la Figura 4.5.

———INFO AQUI——————————————— Ahora, si pinchamos sobre cada panel, podemos observar bla bla bla. (

Más adelante, en la Sección 5.xx **??** se ahonda más en el uso de Kibana, y se compara con Grafana, la herramienta principal de este proyecto.

[6] `https://redis.io/`

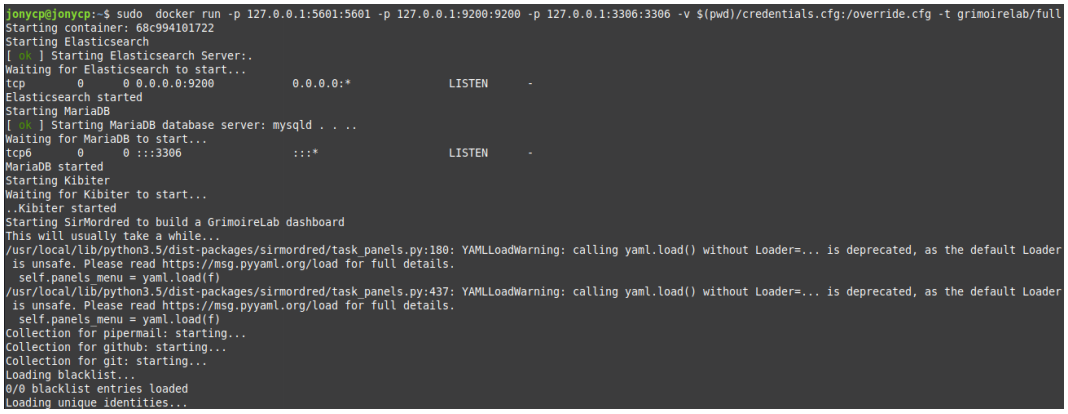

Figura 4.4: Logs mostrados tras el lanzamiento del contenedor de GrimoireLab

4.4. Sprint 2 - Grafana

4.4.1. Instalación

4.4.2. Enlazado

4.4.3. Exploración de índices (ElasticSearch)

4.5. Sprint 3 - Plugins de Grafana

4.6. Sprint 4 - Docker Hub y GrimoireLab

4.7. Sprint 5 - Creando el contenedor

4.7.1. DockerHub Use

USO DE DOCKERHUB

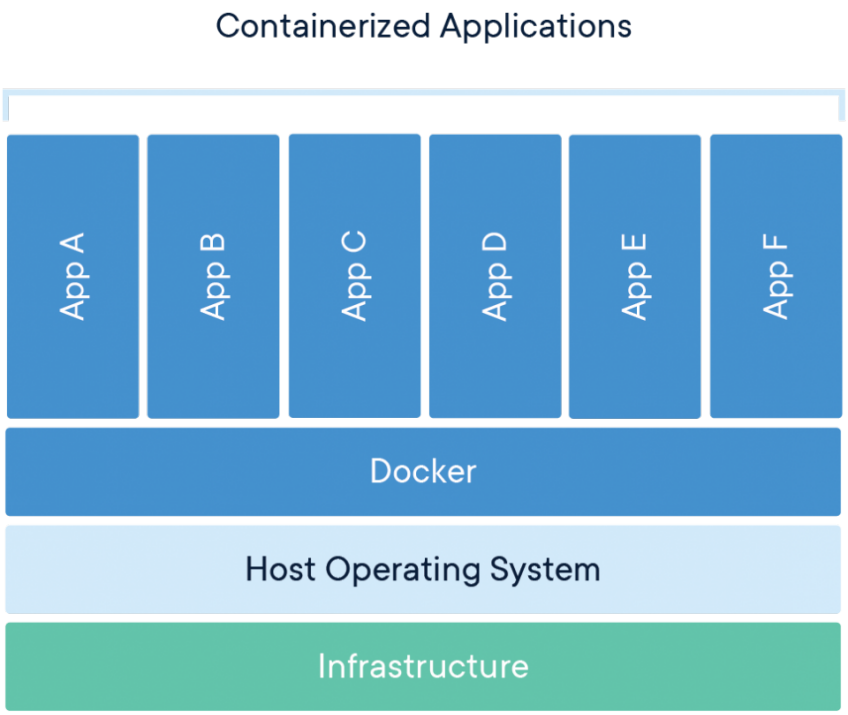

Figura 4.5: Ejemplo de dashboard (PANTALLAZO DE OVERVIEW AQUI)

Capítulo 5

Resultados

En este capítulo se incluyen los resultados de tu trabajo fin de grado.

Si es una herramienta de análisis lo que has realizado, aquí puedes poner ejemplos de haberla utilizado para que se vea su utilidad.

Capítulo 6

Conclusiones

6.1. Consecución de objetivos

Esta sección es la sección espejo de las dos primeras del capítulo de objetivos, donde se planteaba el objetivo general y se elaboraban los específicos.

Es aquí donde hay que debatir qué se ha conseguido y qué no. Cuando algo no se ha conseguido, se ha de justificar, en términos de qué problemas se han encontrado y qué medidas se han tomado para mitigar esos problemas.

6.2. Aplicación de lo aprendido

Aquí viene lo que has aprendido durante el Grado/Máster y que has aplicado en el TFG. Una buena idea es poner las asignaturas más relacionadas y comentar en un párrafo los conocimientos y habilidades puestos en práctica.

1. a
2. b

6.3. Lecciones aprendidas

Aquí viene lo que has aprendido en el Trabajo Fin de Grado/Máster.

1. a

2. b

6.4. Trabajos futuros

Ningún software se termina, así que aquí vienen ideas y funcionalidades que estaría bien tener implementadas en el futuro.

Es un apartado que sirve para dar ideas de cara a futuros TFGs/TFMs.

Apéndice A

Manual de usuario

Índice de figuras