UNIVERSIDAD
REY JUAN CARLOS

MÁSTER EN INGENIERÍA DE LA TELECOMUNICACIÓN

Curso Académico 2017/2018

Trabajo Fin de Máster

Q-DashMan

VISUALIZACIÓN DE DATOS DE DESARROLLO DE SOFTWARE

Autor : Quan Zhou

Tutor : Dr. Jesús María González-Barahona

Dedicado a

mi

II

Agradecimientos

Doy gracias a Bitergia, la empresa que confió en mí justo después de terminar la carrera y siguen confiando en mí, donde hay un ambiente de trabajo inmejorable con unos compañeros que están dispuestos a ayudar en todo lo que necesite. Más que compañeros de trabajo son amigos.

Resumen

Q-DashMan es una aplicación web que sirve para generar dashboards (panel de control) con métricas sobre el desarrollo de un proyecto de software de una manera muy simple. Para ello utiliza Django (como framework de desarrollo de aplicaciones web) y las herramientas de GrimoireLab (un conjunto de herramientas para recoger, analizar y visualizar métricas sobre el desarrollo de software)

Q-DashMan utiliza varias herramientas de GrimoireLab, con el objetivo final de extraer datos de los sistemas donde tiene lugar el desarrollo de software, analizarlos, y finalmente producir un dashboard que muestre las métricas de interés de una forma interactiva. Q-DashMan permite especificar cuáles son esos sistemas para un proyecto concreto, y se encarga de todo el proceso hasta que el dashboard está listo para su uso.

En la Figure 1 se muestra como es la página principal de Q-DashMan y en la Figure 2 el dashboard creado por la aplicación

Las herramientas de GrimoireLab usadas son las siguientes:

- Perceval: Extracción de las múltiples fuentes de datos
- Sortinghat: Afiliación automática de identidades
- Sigils: Visualización de datos
- SirMordred: Orquestación

Por lo tanto, GrimoireLab proporciona las herramientas necesarias. Pero escribir ficheros de configuración para Mordred es tediosos y complicado. Además, obliga a dar acceso a la máquina donde se realizará el análisis, dado que hay que gestionar el fichero de configuración, y ejecutar las herramientas.

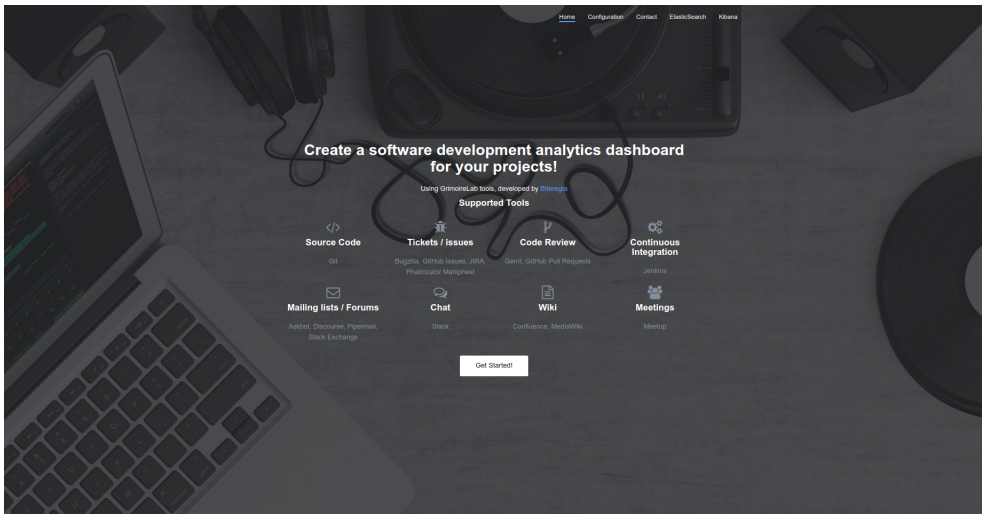

Figure 1: Overview

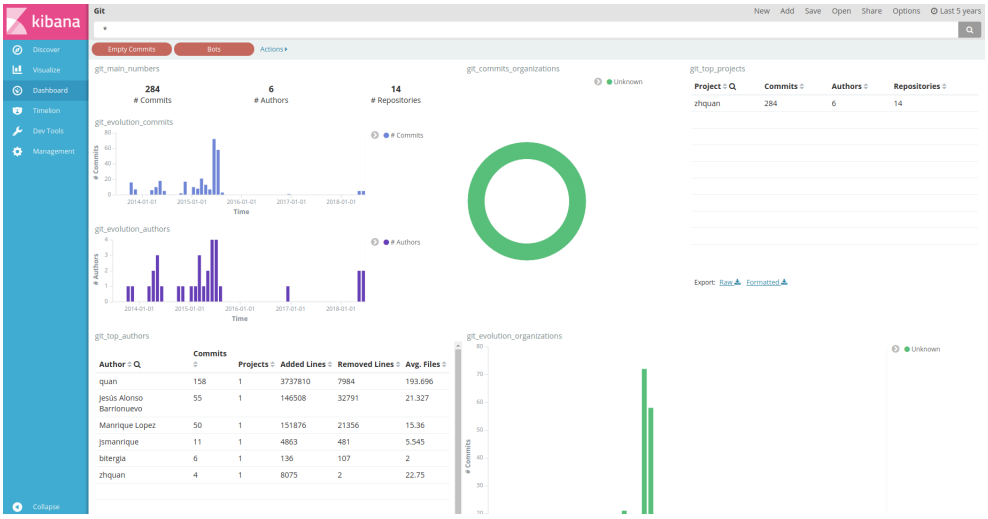

Figure 2: Dashboard

Aquí es donde Q-DashMan ofrece una alternativa: una aplicación web que ofrece la generación de dashboards como servicio. La interfaz de usuario que ofrece es muy simple (comparada con los ficheros de configuración de SirMordred): sólo unos cuantos formularios que habrá que rellenar para especificar las fuentes de datos del proyecto a analizar. El resto es completamente automático: Q-DashMan produce el dashboard ocultando completamente al usuario la complejidad de uso de todas las herramientas de GrimoireLab.

Para construir Q-DashMan se han utilizado tecnologías estándar de desarrollo web, basadas en el framework Django. La herramienta está escrita en Python, usa una base de datos mariadb y se puede desplegar de forma simple utilizando contenedores Docker. La herramienta es fácil de desplegar, y se está probando en la actualidad en entornos de pre-producción, habiéndose probado ya para construir dashboards para decenas de proyectos.

En resumen, Q-DashMan está diseñada de forma que pueda ofrecer la generación de dashboards de desarrollo de proyectos de software como un servicio automático, autoprovisionado por los usuarios, y desatendido.

Abstract

Q-DashMan is a web application used to generate dashboards with metrics about a software's project development in an easy way. For that, we use Django (as development's framework of web applications) and GrimoireLab's tools (several tools to collect, analyze and view metrics about software's development).

Q-DashMan uses several GrimoireLab's tools. Its final goal is to export data from the systems where you develop and analyze the software to produce at the end a dashboard that shows interesting metrics in an interactive way. Q-DashMan let specifying which ones are those systems for a specific project and manages the whole process till the dashboard is ready to work.

In Figure 1 it is showed how is Q-DashMan's main website and in Figure 2 it is showed the dashboard created by the app.

GrimoireLab's tools used are the following:

- SirMordred: Orchestration.
- Perceval: Data collection.
- Sortinghat: Affiliate identities automatically.
- Sigils: Data visualization.

Consequently, GrimoireLab gives you the tools you need. However, writing files set for-Mordred is difficult and boring and it forces you to let the machine access to the place where the analysis is going to happen due to you need to manage the set file and run the tools.

In this part of the process is where Q-DashMan offers an alternative: an app who offers the generation of dashboards as a service. The user interface that it offers is very simple comparted with the files set by SirMordred, you'd only need to complete some files to specify the data

source of the project you're going to analyze. All the rest is completely automatic since Q-DashMan produces a dashboard hiding the difficult use of all GrimoireLab's tools to the user.

We used development web standard technologies to built Q-DashMan based on Django framework. The tool is written in Python, it uses a database MariaDB and it can be deployed in a simple way by using Docker containers. The tool is easy to deploy and it's being used as a trial in environments as preproduction to build dashboards in more than 10 different projects.

As a conclusion, Q-DashMan is designed to offer the generation of development software projects dashboards as an automatic service, generated by users with any management.

Contents

List of Figures

Chapter 1

Introducción

En primer lugar, vamos a hablar un poco de los problemas que hay en la actualidad en el ámbito de la visualización de los datos, las dificultades que se presentan a la hora de querer extraer información de las diferentes fuentes de datos y el masajeo de los mismos, para que a la hora de visualizarlos sean claros y con una interfaz de usuario muy simple e intuitiva.

Por otra parte, hay un producto muy parecido en Bitergia llamado Cauldron[1] que analiza los repositorios de GitHub de una manera muy sencilla: con unos simples clicks y un formulario es capaz de crear un dashboard. No obstante, tenemos que acceder con una cuenta de GitHub para poder empezar a utilizarlo. En la Figure 1.1 vemos la página de inicio de Cauldron y en la Figure 1.2 el dahsboard que genera y si nos fijamos en la parte del menú de la izquierda, vemos los tipos de paneles que hay, que son: *Git*, *Issue Timing*, *Pull Requests*, *PRs Timing*, *Backlog* y *About*. *Kibiter* es un *Kibana* personalizado en el que se ha cambiado el menú que viene por defecto para que acceder a los paneles sea más fácil y más visual a la que ofrece actualmente Kibana.

La diferencia entre Cauldron y Q-DashMan, es que con Q-DashMan no sólo se pueden analizar repositorios de GitHub, si no una gran cantidad de fuentes de datos como Slack, Meetup, MediaWiki, Confluence, etc. Es una aplicación mucho más completa, respecto a la cantidad de fuentes de datos que soporta, pero en cuanto a las gráficas y las métricas que hay en uno y en otro son iguales.

Q-DashMan está creado de una forma modular, es decir, que se pueden añadir nuevas fuentes de datos de una manera simple y en pocos pasos, ya que, cada herramienta que usa

[1] https://cauldron.io/

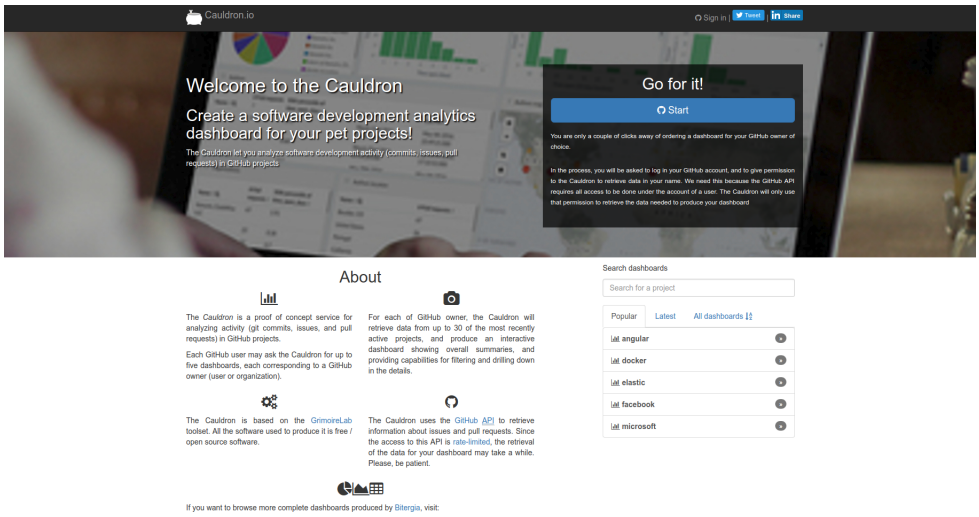

Figure 1.1: Cauldron

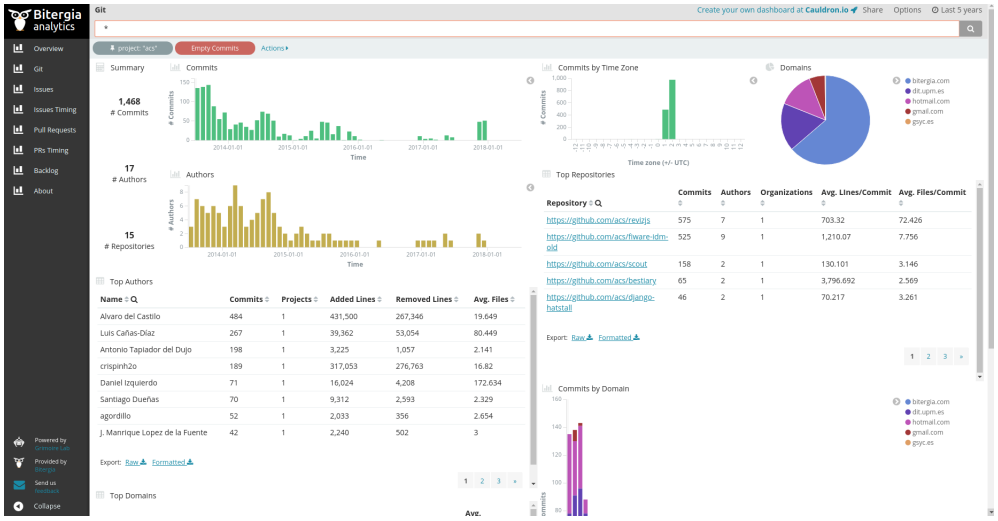

Figure 1.2: Cauldron dashboard

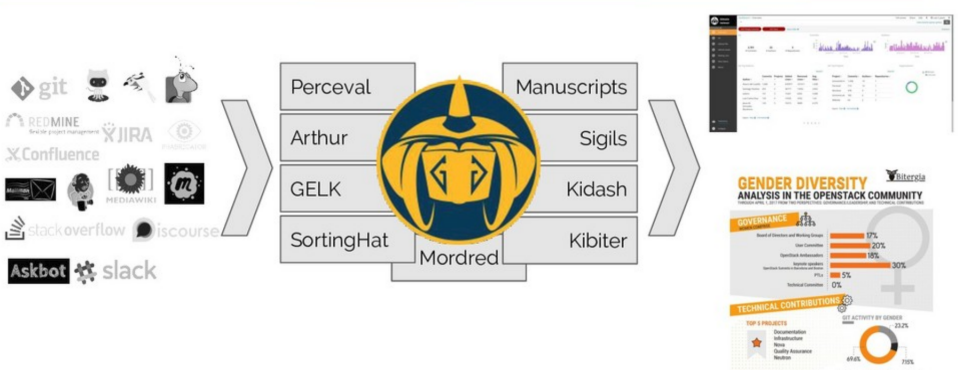

Figure 1.3: GrimoireLab

la aplicación también es modular. Por ejemplo, si queremos añadir una nueva fuente de datos, necesitaríamos desarrollar el backend correspondiente de Perceval para poder extraer esos datos, en Mordred añadir a la lista de fuentes soportadas y en Q-DashMan añadir un formulario más para poder configurar los ficheros de Mordred.

También, con unos cambios muy simples podríamos analizar *GitHub Enterprise*.

Veámos que puede hacer GrimoireLab, cuáles son sus entradas, qué salidas nos da, qué se observa en la Figure 1.3 y qué se explicará en profundidad en cada módulo más adelante. Sin embargo, las partes más importantes para esta aplicación son Perceval, GELK, Sortinghat, SirMordred, Sigils. Dado que no estamos usando todas las herramientas que tiene GrimoireLab, sino tan sólo lo necesario para poder generar dashboards.

1.1 Descripción del problema

¿Qué importancia tienen los datos actualmente para las empresas y las organizaciones?

En la actualidad, los datos cada vez son más importantes en empresas de cualquier sector: las tecnológicas, restaurantes, supermercados, papelerías, inmobiliarias, es decir, a cualquier empresa. Ya que teniendo unos datos adecuados nos podemos adelantar a la competencia y tomar decisiones que harán que nuestra empresa sea más productiva.

Viendo la importancia que tienen los datos, tenemos una pregunta que hacernos. ¿Cómo podemos extraer toda esa información tan dispersa?

Esta pregunta la resolveremos gracias a una herramienta que hemos desarrollado en Bitergia, llamada Perceval. Toda la parte de extracción de información de las diferentes fuentes de datos

la tenemos reunida en esta potente herramienta, la cual va mejorando y añadiendo más fuentes.

Una vez que tenemos los datos, tenemos que enseñarlos de una manera entendible. Extraer los datos es una tarea complicada, pero si no sabemos visualizarlos adecuadamente no nos sirve de nada tener todos esos datos, entonces... ¿Cómo vamos a mostrar esos datos?

Volvemos a usar una herramienta de la empresa llamada Sigils, que nos soluciona este problema. También Sortinghat, que nos muestra a que empresa pertenece cada persona, todas sus cuentas de diferentes herramientas unificadas en una sola, su género, etc. Parece una tarea fácil, pero el 70% de las quejas que tenemos de los clientes, es para que mejoremos su afiliación mediante un fichero que componemos manualmente, debido a que Sortinghat en ocasiones es incapaz de afiliar correctamente, ya sea porque para cada fuente usa un correo electrónico diferente o porque en vez de usar el nombre real usa un apodo, algo muy frecuente en las cuentas de GitHub.

Un ejemplo de estos dashboards que hacemos en la empresa es el dashboard de CHAOSS[2] un proyecto de la Linux Foundation en la que formamos parte. Podemos ver su dashboard en `http://chaoss.biterg.io/`. Si observamos la Figure 1.4, usamos Kibana mientras que Cauldron usa Kibiter, pero tenemos un panel muy importante en el que nos tenemos que fijar y es el de *Data Status*, como se ve en Figure 1.5 que es donde se ve reflejado si el modo incremental está funcionando correctamente o no, porque nos dice cuando fue el primer y el último documento que tiene y también cuando fue la primera y la última ejecución por cada fuente de datos.

Entonces si todo está desarrollado por Bitergia. ¿Qué es lo que nos ofrece esta aplicación? Aunque en Bitergia tenemos todas las herramientas necesarias para poder montar un dashboard, las configuraciones no son sencillas. Por ejemplo, la configuración de Mordred (encargada de la orquestación para que funcionen todas las partes, desde la recogida de datos con Perceval hasta tener los paneles en Kibana usando Sigils) es bastante compleja, porque tiene muchos parámetros y no está claro cuáles son obligatorios, qué parámetros necesita cada sección, ni qué valores tienen que tener.

En esta aplicación, nos vamos a centrar en la configuración de los ficheros para que cualquiera pueda desplegar un dashboard y no sólo los operadores de Bitergia, si no cualquiera que esté interesado. Pero todo ello, ofreciendo una interfaz de usuario muy simple e intuitiva.

[2]https://chaoss.community/

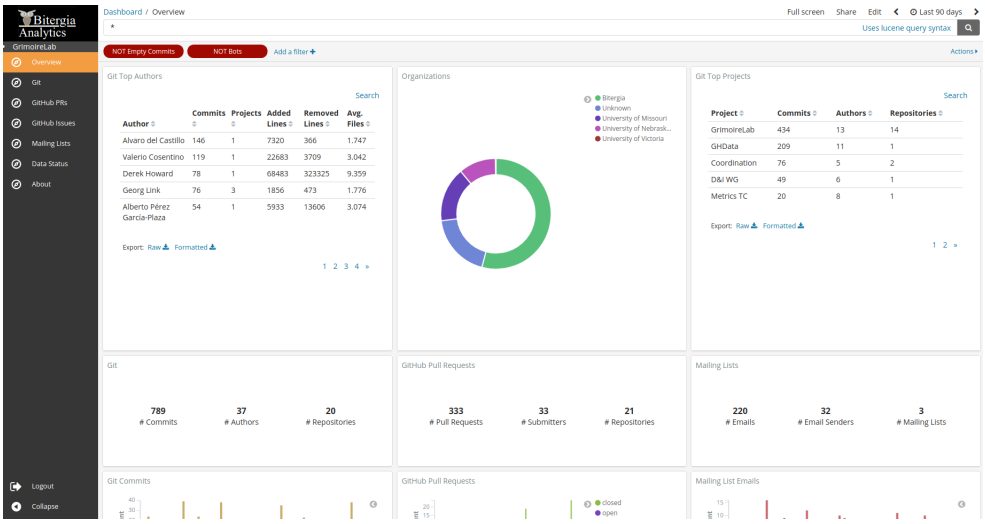

Figure 1.4: Chaoss Overview

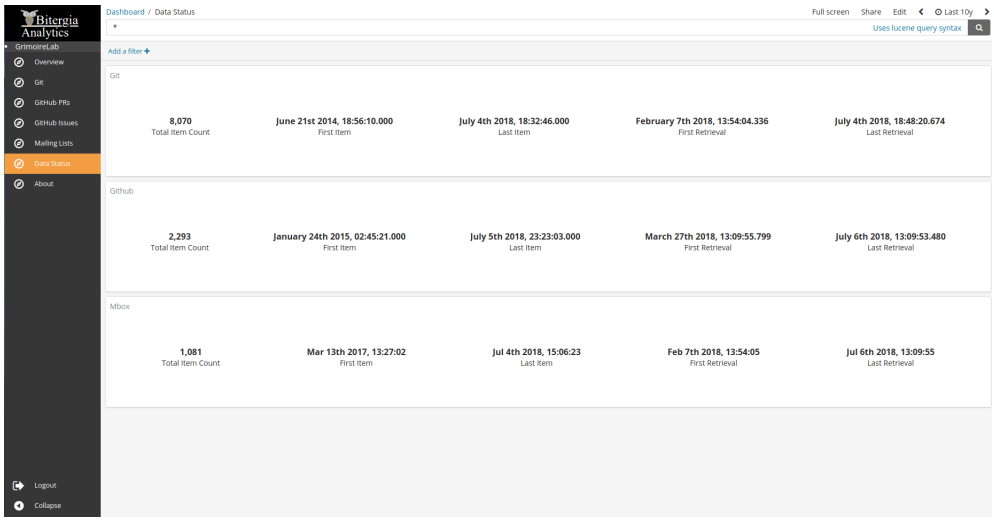

Figure 1.5: Chaoss Data Status

1.2 Objetivo principal

Lo más importante en este proyecto es que el usuario pueda crear su propio dashboard personalizado con una interfaz de usuario muy amigable y en la que rellenar unos formularios muy simples sea suficiente, sin tener que añadir ninguna línea código ni usar el terminal para configurar los ficheros de SirMordred, que es lo que se está haciendo actualmente para poder desplegar un dashboard.

De esta manera, esta aplicación la puede usar cualquier usuario sin necesidad de tener conocimientos técnicos y lo único que tiene que saber son los tipos de fuentes de datos que están soportados. Que son las siguientes:

- Código Fuente: Git
- Tickets / Issues: Bugzilla, GitHub Issues, JIRA, Phabricator Maniphest
- Revisión de Código: GitHub Pull Request
- Integración Continua: Jenkins
- Listas de Correos / Foros: Askbot, Discourse, Pipermail, Stack Exchange
- Chat: Slack
- Wiki: Confluence, MediaWiki
- Meetings: Meetup

En Bitergia podrás encontrar más fuentes de datos y seguimos trabajando en ello para soportar más. Con una analítica más en profundidad, unas configuraciones más avanzadas y en definitiva un servicio más completo, ya que, esta aplicación es una simplificación del producto que se ofrece en Bitergia.

Otro aspecto muy importante es que la aplicación tiene que ser incremental, porque queremos hacer un seguimiento de los proyectos y ver si las decisiones que se han tomado anteriormente son las correctas y cuánto tiempo se necesita para que surja efecto.

1.3 Requisitos

Para que la aplicación pueda ser usada por todos los públicos, la aplicación tiene que ser muy intuitiva.

Por ejemplo, un community manager no tiene porque saber como funcionan las herramientas, ni siquiera necesita entender cual es el proceso. En su mente está que, quiere saber cómo están avanzando los proyectos y quiénes son los principales contribuidores (pero no simplemente el número de commits, sino también la participación en los foros, en los chats, la revisión de código, etc.) Para saber eso lo único que tiene que hacer es introducir esas fuentes en la aplicación con una configuración muy sencilla y en poco tiempo tendrá a su disposición un dashboard con diversos paneles que le ayudará a entender perfectamente el estado de todos los proyectos en un único sitio, además con la posibilidad de crear sus propias gráficas, ya que, es posible que necesite una cierta información que en los paneles por defecto no esté. Es muy difícil o imposible satisfacer las necesidades de todos.

Hay que tener unos conocimientos mínimos de ElasticSearch donde se alojan los datos, MariaDB donde se va a guardar toda la información de las identidades y de Kibana que es quien muestra el dashboard con todos los paneles y gráficas.

Chapter 2

Contexto del proyecto

2.1 Ingeniería del software libre

Todo programa que sea considerado software libre debe ofrecer una serie de libertades. Se resumen en: libertad de usar el programa con cualquier fin, sin necesidad de comunicarlo a los desarrolladores; libertad de estudiar el código fuente del programa y modificarlo adaptándolo a nuestras necesidades, sin necesidad de hacer públicas las modificaciones; libertad de distribuir copias, tanto binarios como código fuente, modificadas o no, gratis o cobrando por su distribución; libertad de modificar el programa y publicar las mejoras para beneficio de la comunidad.

El enfoque sistemático y cuantificable que propone la ingeniería del software siempre tuvo como barreras las propias de las formas en que el software ha sido desarrollado, publicado y distribuido. Aspectos como el formato binario, oscurantismo en modelo de negocio y limitaciones comerciales han impedido validar resultados por parte de equipos independientes.

El reciente auge del software libre aporta novedades a esta ingeniería del software. La implantación de Internet junto con las licencias que fomentan la colaboración en el desarrollo del software, han favorecido a que además del código fuente, se diponga de repositorios de versiones donde observar la evolución del software o listas de correo que reflejan las comunicaciones durante el desarrollo. De estas fuentes puede obtenerse gran cantidad de datos de valor, incluso de forma automatizada.

Varios factores son los aportados a la ingeniería del software tradicional desde ingeniería del software libre:

- Visión temporal incorporada al análisis: necesaria ya que el proceso de creación cambia y su evolución analizada de forma continua proporciona información muy interesante (lenguajes más usados, evolución de colaboradores de un proyecto) destinada a servir de ayuda en la toma de decisiones.

- Análisis a gran escala: dada la inexistencia de impedimentos para ampliar el análisis al conjunto global de los proyectos de software libre gracias a la disponibilidad de la información generada durante su desarrollo. La ingeniería del software libre hace posible evaluar un proyecto dentro de entornos globales y de menor envergadura, ofreciendo información desde distintos puntos de vista, lo cual beneficia en la mencionada toma de decisiones.

En cierto modo, la ingeniería del software libre plantea cuantificar unos parámetros que nos permitan pronosticar con precisión costes, recursos y plazos. En la actualidad el software libre carece de estos métodos, aunque la disponibilidad del código fuente y la información generada durante su desarrollo constituye un enorme potencial para que cambie esta situación. La ingeniería del software pretende también aplicar las cualidades de la ingeniería del software en el desarrollo de proyectos de software libre, de modo que se garantice a los desarrolladores la forma de generar software de calidad siguiendo los paradigmas adecuados. La ingeniería del software pretende aportar resultados objetivos y contrastables acerca de la evolución del software y desterrar así apreciaciones que algunos dan por ciertas. A corto plazo, la ingeniería del software libre tiene por objetivo realizar un análisis completo del desarrollo del software libre permitiendo indagar en los procesos que están involucrados, así como una adaptación de modelos de previsión de costes del estilo de COCOMO en el software propietario. Puede considerarse que el software libre funciona gracias a una ”mano negra”, que hace que el software se genere mágicamente. Es por ello que la ing del soft libre busca comprender los elementos e interacciones que engloba esta laguna de conocimiento denominada”mano negra”.

Se hace imprescindible un análisis de los datos relacionados con el software libre para poder alcanzar los objetivos, previamente descritos, que la ingeniería del software libre se propone. Deberá procurarse que las herramientas empleadas para ello estén disponibles para que grupos independientes puedan verificar los resultados. En este proceso de análisis pueden diferenciarse dos fases. En la primera etapa, un grupo de utilidades independientes entre sí recogen datos

cuantificables del código fuente y otros flujos de información y almacenan los resultados en un formato intermedio, que serán analizados en la siguiente fase. Lo ideal es que esta fase se realice de forma automática. La segunda fase, no tan madura como la anterior, integra programas que toman como entrada los parámetros almacenados en el formato intermedio y se dedican a su análisis, procesado e interpretación. Se han propuesto varios modos de analizar los resultados, de entre las que destacamos: herramientas de análisis de clústers, que a partir de porciones reducidas de datos, agrupan los interrelacionados con el objetivo de categorizarlos; herramientas de análisis estadístico, que simplifican el procesado de grandes cantidades de datos y permiten mostrar gráficamente los resultados; una interfaz web, cuyo fin es, además de proporcionar acceso a los resultados de este gran proyecto, la aplicación de los programas que generan este interfaz a otros proyectos de software libre, de modo que surja una realimentación del proyecto global.

En cuanto a las fuentes a analizar, la que alberga mayor información en potencia es el código fuente. De él pueden extraerse parámetros como tamaño, número de líneas lógicas o físicas, número de desarrolladores, lenguaje de programación, etc. Uno de los estudios pioneros en este campo se encarga de calcular el número de líneas físicas de código de proyectos de software libre y aplicar el modelo COCOMO para obtener conclusiones en torno al coste, tiempo y recursos empleados en el desarrollo del software.

Otras fuentes de interés son aquellas donde se produce intercambio de información entre desarrollCadores, como listas de correo o canales de IRC. De estos últimos aún no se han definido claramente los parámetros a buscar, mientras que de las listas interesa recuperar de cada mensaje del archivo: el nombre y dirección del autor, la fecha, e incluso podría cuantificarse la longitud del mensaje.

Existe otro tipo de fuentes de información compuesto por una serie de herramientas que sincronizan el trabajo de los distintos desarrolladores de un software. Las más comunes son los sistemas de control de versiones, de los cuales obtener conclusiones acerca de la participación de cada desarrollador; y los sistemas de gestión de errores.

Una modalidad más de recopilación, quizás aplicable en un futuro, es la relacionada con la información personal de los desarrolladores, que a día de hoy no suele facilitarse, y que ayudaría a conocer con mayor profundidad la comunidad del software libre. Además, si se dispusiera

de los datos laborales de estos desarrolladores —como proyectos en los que colaboró y horas empleadas— podría establecerse una previsión de costes y recursos para futuros proyectos de software libre.

2.2 Python

Python es un lenguaje de programación interpretado, orientado a objetos, alto nivel y de código abierto pero con una filosofía en la sintaxis que favorezca un código legible.

Se trata de un lenguaje de programación en múltiples plataformas, ya que soporta orientación a objetos, programación imperativa y, en menor medida, programación funcional. Es un lenguaje interpretado, usa tipado dinámico y conteo de referencias para la administración de memoria. Permite varios estilos: programación orientada a objetos, programación imperativa y programación funcional. Otros paradigmas están soportados mediante el uso de extensiones.

Otro objetivo del diseño del lenguaje es la facilidad de extensión. Se pueden escribir nuevos módulos fácilmente en C o C++. Python puede incluirse en aplicaciones que necesitan una interfaz programable.

La comunidad de Python tiene una gran cantidad de librerías, para realizar propósitos de todo tipo.

El 13 de febrero de 2009 se lanzó una nueva versión de Python bajo el nombre clave ”Python 3000” o, abreviado, ”Py3K”. Esta nueva versión incluye toda una serie de cambios que requieren reescribir el código de versiones anteriores.

Para la documentación completa visite `https://docs.python.org/3.1/whatsnew/3.0.html`.

2.3 Django

Django es un framework de desarrollo web de código abierto, escrito en Python.

Django te facilita la creación de sitios web complejos y también pone énfasis en el re-uso, la conectividad y la extensibilidad de componentes, junto con el desarrollo rápido. Python es usado en todas las partes del framework, incluso en configuraciones, archivos y modelos de datos.

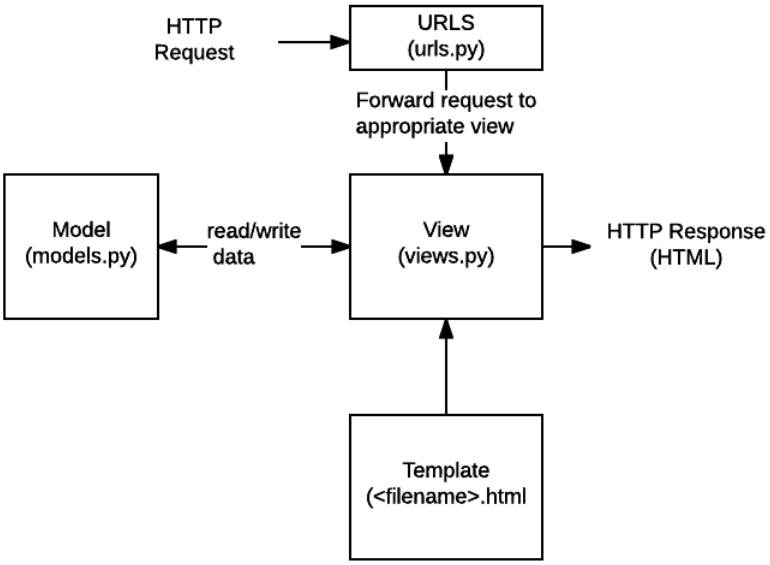

Figure 2.1: Esquema de Django

Las aplicaciones web de Django agrupan el código de esta manera: *urls.py*, *views.py*, *models.py* y los ficheros html para visualizar en los navegadores. Para entender mejor estos ficheros nos fijamos en la Figure 2.1 [1]

- URLs: Procesa las peticiones de cada URL usando un mapeador de URL para redirigir las peticiones HTTP a las vista apropiadas

- View: Una vista es una función de gestión de peticiones HTTP. Dependiendo de las peticiones, puede acceder a los modelos si son necesarios y ofreciendo una respuesta mediante plantillas

- Model: Define la estructura de los datos y la gestiona (añade, borra, modifica y consulta)

- Template: Usa estos ficheros de texto que definen la estructura como una página HTML.

En la aplicación, Q-DashMan usa Django para manejar la parte del servidor. Es importante saber un poco de la arquitectura y como funciona por debajo, aunque sea muy superficial.

[1] https://developer.mozilla.org/es/docs/Learn/Server-side/Django/Introducción

2.4 ElasticSearch

Elasticsearch es un servidor de búsqueda basado en Lucene, el cual provee un motor de búsqueda de texto completo, distribuido y con capacidad de multi-tenencia con una interfaz web RESTful y con documentos JSON. Elasticsearch está desarrollado en Java y está publicado como código abierto bajo las condiciones de la licencia Apache.

Elasticsearch utiliza Query DSL (Lenguaje de dominio específico) para realizar las consultas a los documentos indexados. Es un lenguaje sumamente flexible y de gran alcance, además de simple, que permite conocer y explorar los datos de la mejor manera. Al ser utilizado a través de una interfaz de tipo JSON, las consultas son muy sencillas de leer y, lo más importante, de depurar.

Sobre la base de su estructura y componentes, las consultas se componen de dos cláusulas: ”Leaf Query Clauses” y ”Compound Query Clauses”. La primera hace referencia a aquellas consultas que tienen operaciones como ”match”, ”term” o ”range”, que devuelven un valor específico solicitado y las segundas son una combinación de todos los apartados de la primera, una manera de realizar consultas ”conjuntas” para obtener información más compleja y detallada.

2.5 Kibana

Kibana es una herramienta open-source de Elastic, que nos permite visualizar y explorar datos que se encuentran indexados en ElasticSearch.

Kibana ofrece una interfaz muy potente que permite crear dashboards a medida, seleccionar rangos, compartirlos y guardarlos. Estos dashboards se crean directamente en su UI, lo que hace que sea fácil de crear y muy visual.

Tiene un número elevado de gráficas como las de tarta, barra, áreas, mapa, tabla, etc. Todas estas gráficas están interconectadas, es decir, si aplicamos un filtro o una búsqueda en una de ellas, se verán afectadas todas ellas, incluso también puede hacer que ese filtro se mantenga en todos los dashboards que hayas creado.

Veamos un ejemplo de un dashboard en Kibana con diferentes visualizaciones en la Figure 2.2[2]

[2]https://www.elastic.co/products/kibana

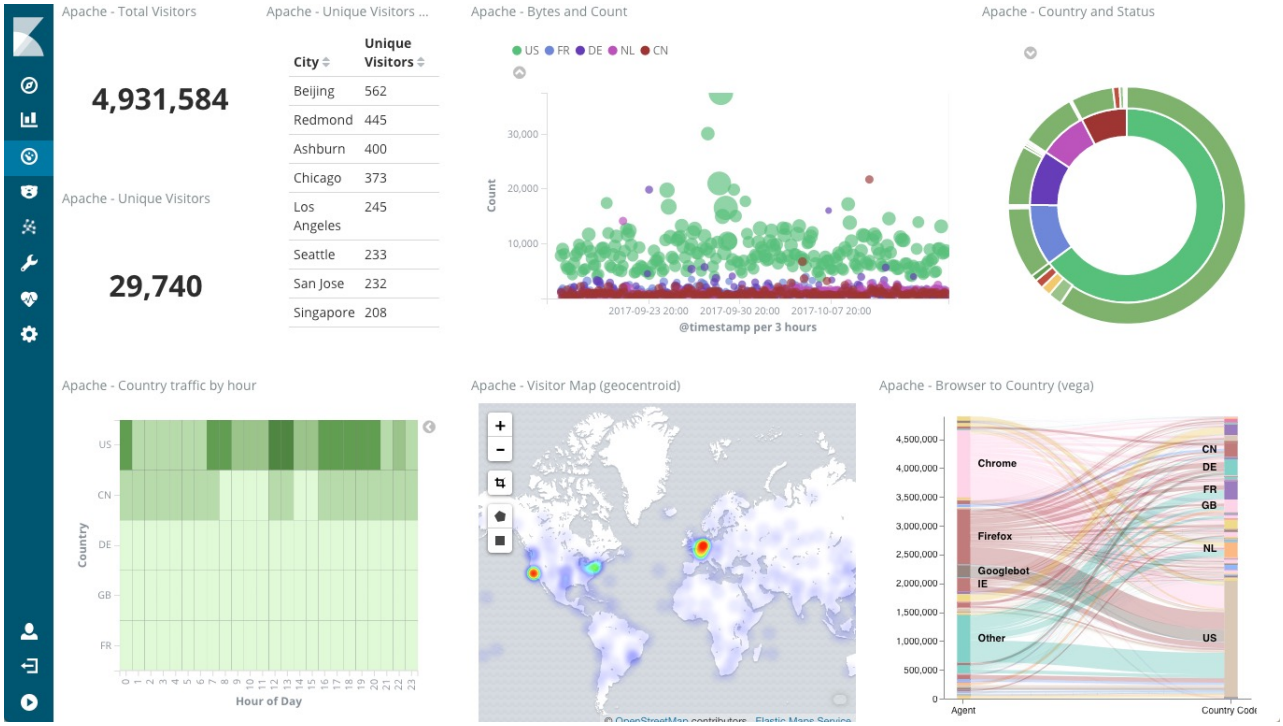

Figure 2.2: Dashboard en Kibana

Se usará exclusivamente Kibana para la visualización de los datos, aunque en Bitergia han creado su propio Kibana llamado Kibiter.

2.6 HTML

HTML, siglas de HyperText Markup Language (lenguaje de marcas de hipertexto), hace referencia al lenguaje de marcado para la elaboración de páginas web y es un estándar que sirve de referencia para la elaboración de las mismas en sus diferentes versiones, definiendo una estructura básica y un código (denominado código HTML) para la definición del contenido de una página web, como el texto, las imágenes y los videos, entre otros. Es un estándar a cargo de la W3C, organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación.

El lenguaje HTML basa su filosofía de desarrollo en la referenciación. Para añadir un elemento externo a la página (imagen, vídeo y script, entre otros), éste no se incrusta directamente en el código de la página, sino que se hace una referencia a la ubicación de dicho elemento mediante texto. De este modo, la página web contiene sólo texto mientras que recae en el nave-

gador web (interpretador del código) la tarea de unir todos los elementos y visualizar la página final. Al ser un estándar, HTML busca ser un lenguaje que permita que cualquier página web escrita en una determinada versión, pueda ser interpretada de la forma estándar por cualquier navegador web actualizado.

2.7 CSS

Hoja de estilo en cascada o CSS (siglas en inglés de cascading style sheets) es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML o XML2 (y por extensión en XHTML).

La información de estilo puede ser definida en un documento separado o en el mismo documento HTML. En este último caso podrían definirse estilos generales en la cabecera del documento o en cada etiqueta particular mediante el atributo $< style >$.

2.8 Bootstrap

Bootstrap son plantillas echas en base de HTML5, CSS3 y JQuery en javaScritp que nos ofrece una apariencia bastante atractiva y que a su vez es compatible con diversas plataformas (móviles, tabletas y ordenadores).

Esta herramienta tiene diversas plantillas orientadas a determinados tipos de páginas o aplicaciones de manera orientativa y es bastante flexible, ya que puede cambiar de aspecto como se quiera para poder ser personalizado al gusto de cada uno. Aunque éste aspecto no es su punto fuerte, ya que existen en la red infinidad de plantillas echas con HTML5, CSS3 igual o mejor, éste tiene una plantilla altamente adaptada para diferentes plataformas, lo que lo vuelve mucho más atractivo. Antes de mostrar la página, primero reconoce el tipo de plataforma a la que se está accediendo en ese momento y dependiendo de cada una de ellas nos aparecerá de una forma u otra pero con una apariencia similar, con algunos detalles necesarios para su buen funcionamiento. Esto hace que bootstrap sea usado por tantas personas.

Veamos algunos ejemplos de plantilla de esta biblioteca en Figure 2.3 y Figure 2.4

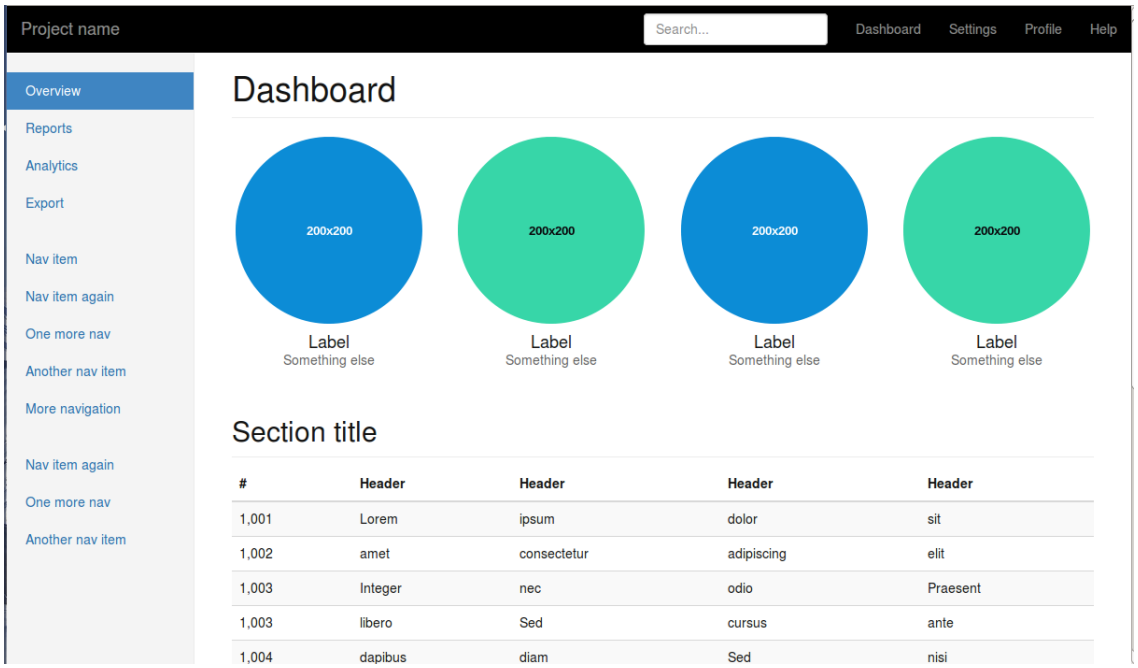

Figure 2.3: Bootstrap Dashboard

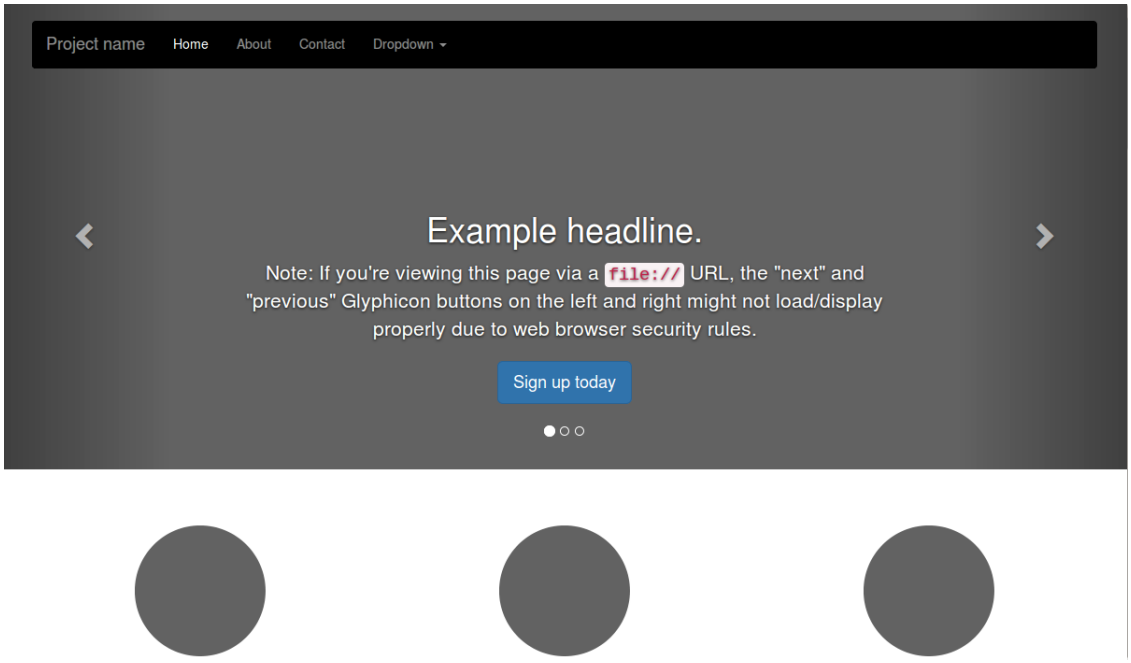

Figure 2.4: Bootstrap Carousel

2.9 JQuery

jQuery es una biblioteca de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con AJAX a páginas web. Siendo JQuery la biblioteca de JavaScript más utilizada.

jQuery es un software libre y de código abierto, permitiendo su uso en proyectos libres y privados. Al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes resultados en menos tiempo y espacio.

2.10 GitHub

GitHub es una plataforma para alojar proyectos utilizando el sistema de control de versiones Git. Donde está alojado este proyecto `https://github.com/zhquan/Q-DashMan`.

Una herramienta verdaderamente útil para los desarrolladores. Puedes observar como evolucionan los proyectos, las nuevas versiones, volver a las versiones anteriores si es necesario. Muchas de las bibliotecas que se han usado para hacer este proyecto también están alojados sus códigos fuentes en `http://github.com`

GitHub también nos puede servir una página html creando una rama llamada *gh-pages*, para que nos pueda mostrar la página web que hemos desarrollado. De esta manera podemos ver nuestro proyectos en la red con la URL que te proporciona GitHub, según tu nombre de usuario y en el repositorio en el que se encuentre *"(user-name).github.io/(repo-name)/"* como por ejemplo `http://zhquan.github.io/TFG/` donde puedes ver mi trabajo fin de grado.

Chapter 3

Desarrollo e implementación

En este capítulo explicaremos como se ha desarrollado la aplicación y el por qué.

Después de varios años trabajando en el equipo de operaciones de Bitergia, mi labor me obliga a estar informado de la vanguardia de las tecnologías que se usan para generar los dashboards y como generarlos. En mi trabajo diario una de las tareas que tengo que hacer es desplegar nuevos entornos para los clientes como dashboards y crear informes a base de esos dashboards. También hay que mantenerlos, que todos los datos estén siempre actualizados, y si no lo estuvieran, habría que detectar los errores y arreglarlos o si es algo del software pasarlo al equipo de desarrollo. Obviamente, todos los entornos tienen que estar en la última versión estable del producto, ya que, es imposible mantener una infraestructura en la que cada uno tenga una versión diferente.

Después de una larga reflexión acerca de cómo se podría desplegar de una forma mucho más rápida, mucho más sencilla, mucho más intuitiva, que cualquiera de la empresa sea capaz de hacerlo. No obstante, ahora mismo en la empresa se hace en varias fases que no son muy intuitivas como la configuración de los ficheros de SirMordred, donde se tienen que crear esos ficheros, los permisos que tienen que tener cada uno de ellos, cual es la versión estable de cada componente y debido a eso, sólo el equipo de operaciones sabemos el procedimiento entero.

Una de las peticiones que tenemos nos viene directamente desde dentro de la propia empresa, que es la de crear dashboards específicos no para el cliente, sino para nosotros mismos, como en el caso del equipo de analítica que necesitan un dashboard con ciertos datos para poder crear nuevos paneles o simplemente porque han sacado una versión nueva y necesitan un sitio de pruebas. También, en el caso del equipo de marketing necesitan un dashboard para hacer

presentaciones, para un artículo o simplemente para hacer publicidad.

Después de todo esto, era necesario crear una aplicación que cumpliera con todos estos requisitos, y así es como empecé a desarrollar Q-DashMan para que el resto del equipo pudiera montar sus propios dashboards.

No obstante, para una óptima comprensión necesitaremos previamente explicar cómo funciona la aplicación por dentro, es decir, cómo funciona GrimoireLab que es el motor de esta aplicación y cómo están conectados. Vemos que todo esta orquestado por SirMordred que lee unos ficheros de configuración y se encarga de ejecutar GELK con todo lo necesario para que cada componente funcione ordenadamente.

3.1 GrimoireLab

GrimoireLab es un software de código abierto para el análisis de desarrollo de software, el cual permite recoger datos de muchos tipos de sistemas con información relacionada al desarrollo de software, y produce análisis y visualizaciones de los mismos.

GrimoireLab soporta más de 20 diferentes fuentes de datos, desde repositorios de git o proyectos GitHub, para *trackear* issues como Jira o Bugzilla, incluyendo sistemas de mensaje como IRC, Slack o listas de correo, y otros sistemas como StackOverflow o Jenkins. Con las herramientas de GrimoireLab puedes recoger los datos, guardarlos en bases de datos, analizarlos y producir un dashboard con visualizaciones interactivas de los datos resultantes.

Veamos que estructura tiene GrimoireLab y donde está situado Q-DashMan (Figure 3.1) y cómo están conectados entre sí todos los componentes. Como se puede observar Q-DashMan se encarga de las configuraciones que necesita SirMordred.

3.1.1 GrimoireLab-Perceval

Para recoger esta información de las diversas fuentes de datos vamos a usar Perceval, desarrollado principalmente por Bitergia, lo que nos ofrece la extracción de más de 30 fuentes de datos como Git, GitHub, Confluent, Discourse, Slack, etc. Debido a su continuo crecimiento, cada vez se van añadiendo más. Esta herramienta permite solventar la parte más difícil, ya que al obtener información de tantas fuentes de datos necesitaríamos una gran cantidad de tiempo y posiblemente no lo hagamos correctamente, así, Perceval hace posible que con sólo unos

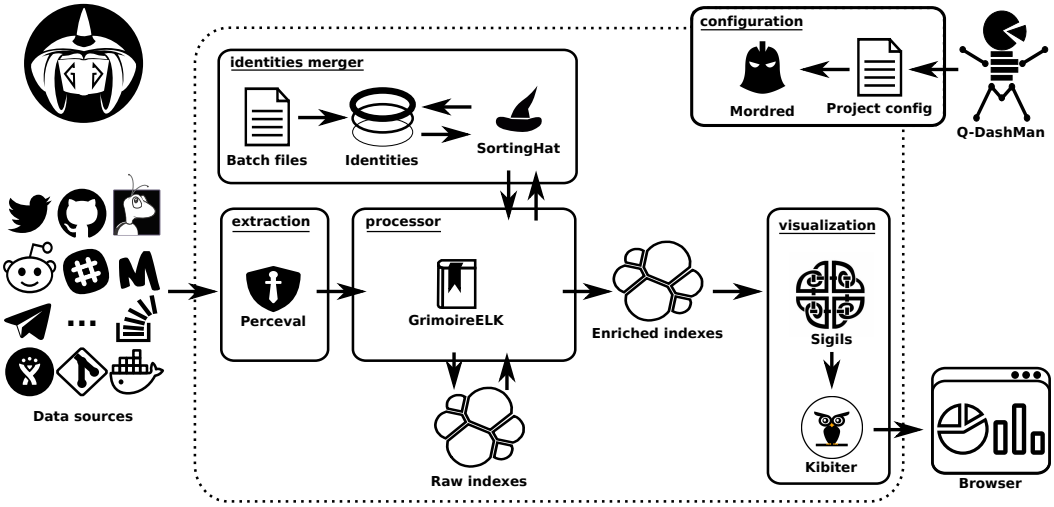

Figure 3.1: GrimoireLab with Q-DashMan Schema

parámetros podamos sacar toda la información que queramos. Para más información visite el repositorio en GitHub `http://perceval.readthedocs.io`

3.1.2 GrimoireLab-Sortinghat

Si no tenemos ninguna información sobre la empresa en cuestión, es bastante complicado afiliar un desarrollador. Por ello, vamos a usar una herramienta llamada Sortinghat, lo suficientemente potente para esta función, ya que tiene muchos métodos para afiliar. Vamos a ver sólo una parte de lo que puede hacer. Para más información visite el repositorio en GitHub `https://github.com/chaoss/grimoirelab-sortinghat`.

- Género: Usando la biblioteca Genderize.io para determinar el género de cada desarrollador.

- Afiliación: Afilia automáticamente a los desarrolladores según su dominio de email.

- Matching: Se puede hacer matching según su email, nombre de usuario (GitHub, Bugzilla, etc) y/o por nombre.

- Mamual: Todo lo anterior se puede cambiar de una forma manual, dado que muchas personas usan su correo personal como gmail, outlook y es imposible hacerlo automático.

3.1.3 GrimoireLab-ELK

Una vez que tenemos las herramientas como Perceval y Sortinghat, ¿quién se encarga de escribir todos esos datos en el ElassticSearch?. El equipo de Bitergia también nos proporciona otra herramienta, llamado GELK (GrimoireLab-ELK). Vamos a ver que es lo que hace GELK, pues fundamentalmente se encarga de crear los índices crudos y enriquecidos:

- Recogida: llama a Perceval para que le dé los documentos y los escriba en el Elastic-Search como índice crudo (raw_index).

- Enriquecido: Llama a Sortinghat para que le dé información sobre la afiliación junto con el índice crudo que selecciona los datos útiles y/o añade información extra para escribirlo en el ElasticSearch nuevamente, en otro índice llamado enriquecido (enriched_index)

3.1.4 GrimoireLab-Sigils

Si tenemos una gran cantidad de datos, pero no sabemos sacarles provecho, no nos sirve de nada lo que tenemos, pues necesitamos aplicar ciertos filtros a los datos y realizar las cuentas pertinentes para poder entenderlas. En este caso, vamos a usar la herramienta llamada Sigils que se encarga de configurar Kibana, crear los alias donde se apuntarán los índices enriquecidos y las visualizaciones para las fuentes de datos. Éstas están separados por paneles y es posible que una de ellas contenga más de uno como git que tiene Overview, Backlog y Timming.

3.1.5 GrimoireLab-SirMordred

Por último, necesitamos que SirMordred orqueste todas las herramientas que tenemos para que funcione a la perfección, cogiendo sus ficheros de configuraciones, setup.cfg, y el de proyectos, projects.json.

SirMordred tiene cuatro fases principales que son la recogida, el enriquecimiento, las identidades y los paneles. Éstas se configuran en el fichero *setup.cfg*. Para tener una idea completa vamos explicar cual es el flujo completo, es decir, con las cuatro fases activadas.

Lo primero que hace SirMordred es la fase de recogida y para ello manda una petición a GELK para que haga llamadas a Perceval con el objtivo de extraer los datos de las diferentes fuentes y escribirlos en ElasticSearch donde se identifican como índices crudos. Una vez que

termina la primera fase empieza la de identidades, de la que se encarga Sortinghat de ello llamada por GELK. Pasamos a la fase de enriquecimiento, donde usando los datos de la primera fase se realiza un análisis para cada fuente de datos con el fin de limpiar los índices, ya que en los índices crudos residen datos inútiles que se deben tratar, y esto, junto con la información que recibe la segunda fase(Sortinghat), crea los índices enriquecidos en ElasticSearch. Por último, GELK llama a Sigils para que cree los paneles y las gráficas de cada fuente de datos y los visualice el dashboard en un Kibana.

3.2 Componer el fichero de proyectos

Una de las dificultades que se presentan es que cuando queremos analizar una organización de GitHub tenemos muchísimos repositorios y añadir esos repositorios uno a uno manualmente es tedioso. Este fue el detonante de crear una herramienta que recoja solo esos repositorios, pasándole únicamente el nombre de la organización para ahorrar el máximo tiempo posiblen la configuración del fichero de proyectos, ocurriendo lo mismo cuando pensamos en slack y meetup.

Para facilitarle el trabajo al operador y a cualquiera que quiera generar un dashboard, generé varias herramientas que con sólo meter unos parámetros producirían todos los repositorios, canales, grupos, etc.

El fichero de proyectos tiene el siguiente formato

```
{
    "<project1>":{
        "git": [
            "https://github.com/<org1>/repo1.git",
            "https://github.com/<org1>/repo2.git"
        ],
        "github": [
            "https://github.com/<org1>/repo1",
            "https://github.com/<org1>/repo2"
        ],
        "slack": [
            "ASDKAPWJ",
            "GJEODPRJ"
```

```
            ]
        },
        "<project2>": {
            "git": [
                "https://github.com/<org2>/repo1.git",
                "https://github.com/<org2>/repo2.git"
            ],
            "github": [
                "https://github.com/<org2>/repo1",
                "https://github.com/<org2>/repo2"
            ],
            "meetup": [
                "group1",
                "group2",
                "group3"
            ]
        }
    }
```

Vamos a detallar las herramientas creadas para que está configuración sea muy sencilla.

3.2.1 GitHub

Para obtener todos los repositorios de una organización en GitHub la herramienta utilizada se llama 'gitHubSync.py'. Esta biblioteca permite sacar todos los repositorios bajo una organización o usuario y meterlos en un diccionario con el siguiente formato:

```
{
    "<organization>":{
        "git": [
            "https://github.com/<organization>/repo1.git",
            "https://github.com/<organization>/repo2.git"
        ],
        "github": [
            "https://github.com/<organization>/repo1",
            "https://github.com/<organization>/repo2"
        ]
    }
}
```

Existe la opción de añadir los repositorios forkeados[1], aunque por defecto son ignorados porque generalmente queremos analizar nuestros proyectos.

Otra cosa factible es cargar el fichero de proyectos desde un repositorio y actualizarlo, ya que muchas veces queremos añadir más proyectos o borrar los mismos.

¿Cómo se usa la biblioteca GitHubSync?
Importamos la librería.

```
In [1]: from githubsync import GitHubSync
```

Podemos usar un token de GitHub ó si son pocas llamadas a la API lo podemos dejar vacío.

```
In [2]: token = ""
In [3]: github = GitHubSync(token)
```

Esta biblioteca tiene unos cuantos métodos, pero sólo nos vamos a centrar en uno, el 'sync'. Este método recibe dos parámetros y devuelve un diccionario con el formato de proyectos de SirMordred. El primer parámetro es un diccionario, porque si ya tenemos un diccionario con varios proyectos podemos seguir añadiendo, y el segundo parámetro es el nombre de la organización o usuario que hay en GitHub como por ejemplo 'zhquan', que es el mio propio. Este proceso llevado acabo en este método, nos devolverá un diccionario con todos los repositorios separados en git y GitHub. La razón de esta división es que podamos analizar los 'commits' de git y los 'issues', 'pull request' de GitHub

```
In [4]: projects = {}
In [5]: github.sync(projects, "zhquan")
Out[5]:
{'zhquan': {'git': ['https://github.com/zhquan/DAT-Mashup.git',
   'https://github.com/zhquan/DAT-PrincesGame.git',
   'https://github.com/zhquan/ISI-Practica-7.git',
   'https://github.com/zhquan/ISI-Practica-ChatAlien.git',
   'https://github.com/zhquan/ISI-Practica-leaderboard.git',
   'https://github.com/zhquan/ISI-Practica8.git',
   'https://github.com/zhquan/owl.git',
   'https://github.com/zhquan/PFG.git',
   'https://github.com/zhquan/Practica-3.git',
   'https://github.com/zhquan/PST-PeerChat.git',
   'https://github.com/zhquan/Q-DashMan.git',
   'https://github.com/zhquan/qz-dashboard.git',
```

[1]Repositorios no originales que son copias de otros

```
 'https://github.com/zhquan/SAT-MiRevista.git',
 'https://github.com/zhquan/TFG.git',
 'https://github.com/zhquan/videochat.git'],
'meta': {'title': 'zhquan'},
'github': ['https://github.com/zhquan/DAT-Mashup',
 'https://github.com/zhquan/DAT-PrincesGame',
 'https://github.com/zhquan/ISI-Practica-7',
 'https://github.com/zhquan/ISI-Practica-ChatAlien',
 'https://github.com/zhquan/ISI-Practica-leaderboard',
 'https://github.com/zhquan/ISI-Practica8',
 'https://github.com/zhquan/owl',
 'https://github.com/zhquan/PFG',
 'https://github.com/zhquan/Practica-3',
 'https://github.com/zhquan/PST-PeerChat',
 'https://github.com/zhquan/Q-DashMan',
 'https://github.com/zhquan/qz-dashboard',
 'https://github.com/zhquan/SAT-MiRevista',
 'https://github.com/zhquan/TFG',
 'https://github.com/zhquan/videochat']}}
```

Como se puede observar, es muy sencillo recoger todos los repositorios bajo una organización o un usuario.

3.2.2 Slack

Este script 'get_channel_slack.py' obtiene de manera muy sencilla todos los canales a los que te hayas suscrito en Slack. Sólo tiene que tener el api-token y decir en que proyecto lo quiere guardar. ¿Por qué en este caso necesitamos el nombre del proyecto y en el caso de GitHub no?. La respuesta es que en GitHub se usa el nombre de la organización como nombre del proyecto, pero en Slack no es necesario introducir el nombre del grupo al que perteneces para ponerlo como nombre del proyecto, ya que el api-token contiene toda esa información. Como podrás imaginar el formato será igual:

```
{
    "<proyecto>":{
        "slack": [
            "ASDQWRGC",
            "DGJKWIWJ",
```

```
                "DVOWOIEG"
            ]
        }
    }
```

Veamos como se usa este script.

```
$ python3 get_channel_slack.py -h
usage: get_channel_slack.py [-h] [-t TOKEN] [-j JSON] [-p PROJECT]

optional arguments:
  -h, --help            show this help message and exit
  -t TOKEN, --token TOKEN
                        slack token
  -j JSON, --json JSON  projects.json path
  -p PROJECT, --project PROJECT
                        project name
```

Según nos muestra la ayuda necesitamos el api-token, el nombre del proyecto al que vamos a añadir los canales y un fichero JSON de proyectos en el formate de SirMordred.

3.2.3 Meetup

En el caso de los grupos de Meetup es bastante similar a los canales de Slack, porque también se necesita poner a que proyecto quiere que se añada, pero en vez del api-token es un topic y en este caso estamos parseando directamente un html con 'BeautifulSoup' usando el script llamado 'get_group_meetup.py'. Como podrás imaginar el formato será igual:

```
    {
        "<proyecto>":{
            "meetup": [
                "group1",
                "group2",
                "group3"
            ]
        }
    }
```

Veamos como usar este script: Importamos la función

```
In [1]: from get_group_meetup import get_groups
```

Y le pasamos el topic que queremos para sacar los nombres de los grupos

```
In [2]: get_groups("python")
Out[2]:
['Hackers-and-Founders',
 'designersgeeks',
 'SF-Data-Science',
 'gdg-silicon-valley',
 'nycpython',
 'Women-Who-Code-SF',
 'codecrewny',
 'WomenWhoCodeNYC',
 'sfpython',
 'Designers-Geeks-NY',
 'Silicon-Valley-Machine-Learning',
 'Cognitive-Toronto',
 '106miles',
 'Women-Who-Code-DC',
 'pydelhi',
 'opensourceblr',
 'bostonpython',
 'software',
 'PyData-London-Meetup',
 'she-codes-tlv',
 ...
 ...
]
```

3.2.4 Manual

Las herramientas anteriores nos ayudan a simplificar la composición del fichero de proyectos, pero muchas fuentes de datos no necesitan un script para sacar los repositorios o grupos, debido a que sólo se necesita un URL como es el ejemplo de phabricator, mediawiki, confluence, etc. Para todos ellos, tenemos esta opción en la que añades uno a uno manualmente, pero de una manera muy sencilla. Con saber a que proyecto lo quiere añadir, cómo se llama la fuente de datos y el URL. Como se ve en la figura **??** es un formulario muy simple.

3.3 Componer el fichero de configuración de SirMordred

Otra dificultad es el fichero de configuración de SirMordred, que es poco intuitivo y muy lioso, ya que son muchos parámetros los que tenemos que rellenar, algunos obligatorios que podemos pasar desapercibidos.

Este fichero es mucho menos intuitivo que el del proyecto, porque no sólo configura una cosa, sino que configura lo que tiene que hacer cada componente (GELK, Sigils, Perceval, Sortinghat). Como es de esperar, tenemos muchos parámetros y lo malo es que, no sabemos cuales son obligatorios y que valores tienen que tener.

El fichero de configuración tiene el siguiente formato:

```
[general]
short_name = quan
update = true
min_update_delay = 720
debug = true
logs_dir = /home/bitergia/logs

[projects]
projects_file = /home/bitergia/conf/sources/projects.json

[es_collection]
url = http://elasticsearch:9200

[es_enrichment]
url = http://elasticsearch:9200
autorefresh = true

[sortinghat]
host = mariadb
user = root
password =
database = quan_sh
load_orgs = false
orgs_file = /home/bitergia/conf/orgs_file
autoprofile = []
matching = [email]
```

```
sleep_for = 50
bots_names = []
unaffiliated_group = Unknown
affiliate = true

[phases]
collection = true
identities = true
enrichment = true
panels = true

[git]
raw_index = git_tfm
enriched_index = git_tfm_enriched
latest-items = true
studies = [enrich_demography]

[github]
raw_index = github_tfm
enriched_index = github_tfm_enriched
api-token = xxxxxxx
sleep-for-rate = true
```

Esta aplicación facilita en gran medida el proceso debido a que te ofrece la posibilidad de rellenar un formulario en el que los parámetros están predefinidos y usted únicamente tiene que completarlo ofreciendo sus valores personalizados o aquellos que se le proporcionan por defecto. Además, nos indica aquellos que se deben rellenar de forma obligatoria.

Se han separado en otro formulario la configuración de las fuentes de datos, ya que todos tienen la misma estructura y tenerlo dividido es bastante sencillo y queda mucho más claro para el usuario a la hora de añadir esa configuración.

A continuación, les mostraremos de qué manera se puede configurar este fichero con la aplicación. En el caso de que ya lo tengamos guardado, podemos importarlo directamente como se ve en la figura Figure 3.2. Si no tenemos ningún fichero hemos de configurarlo desde cero.

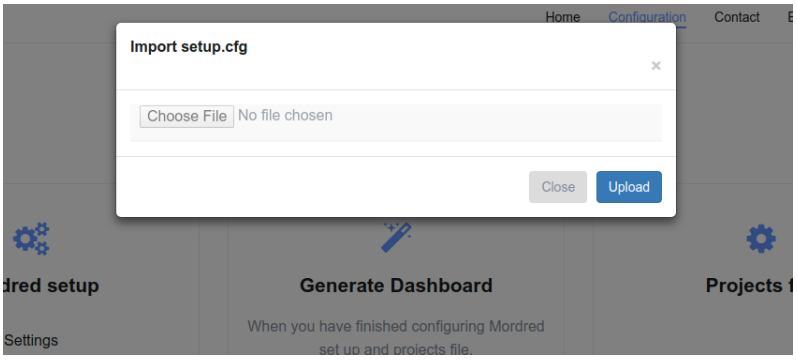

Figure 3.2: Import SirMordred Setup

1. Hacemos click en 'General Settings' y nos aparecerá el siguiente formulario, como se observa en la figura Figure 3.3 podemos ver que tenemos algunos valores que están rellenados debido a que esos parámetros son obligatorios y normalmente siempre tienen esos valores y el problema es que cuando tenemos parámetros que no cambian nunca tendemos a olvidarlos porque prestamos atención a los que son diferentes.

2. Una vez que tenemos esta parte rellenada es hora de añadir fuentes de datos. Haciendo click en 'Add Data Source', como se vé en la figura Figure 3.4, vemos que no tiene tantos paŕametros como en Figure 3.3 debido a que en este formulario tenemos tres parámetros que son obligatorios para todas las fuentes de datos: 'Data Source', 'Raw Index' y 'Enriched Index'. El resto de parámetros depende de cada fuente de datos, pero si en algún momento queremos borrar alguna fuente de datos es incluso más sencillo, pues sólo tiene que hacer click en 'Remove Data Source' Figure 3.5, introducir el nombre de la fuente de datos y se borrará. Una vez que hayamos terminado pueden ver la configuración entera haciendo click en 'Show' Figure 3.6 para ver si se le ha olvidado algo.

3.4 Front-End

Para el front-end se usa bootstrap para crear una pagina web elegante y moderna. El mismo producto con una buena presentación te puede hacer vender 10 veces más y de allí la importancia de la figura del marketing.

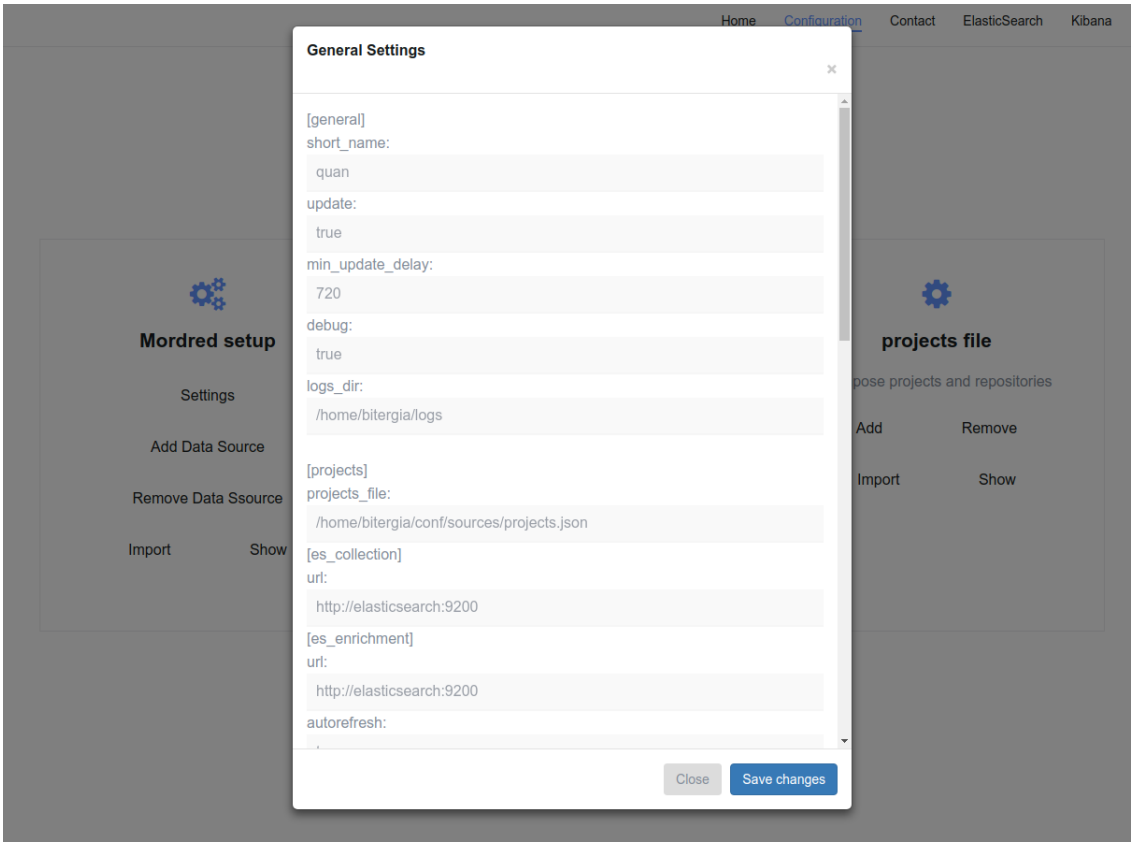

Figure 3.3: SirMordred Setup Settings

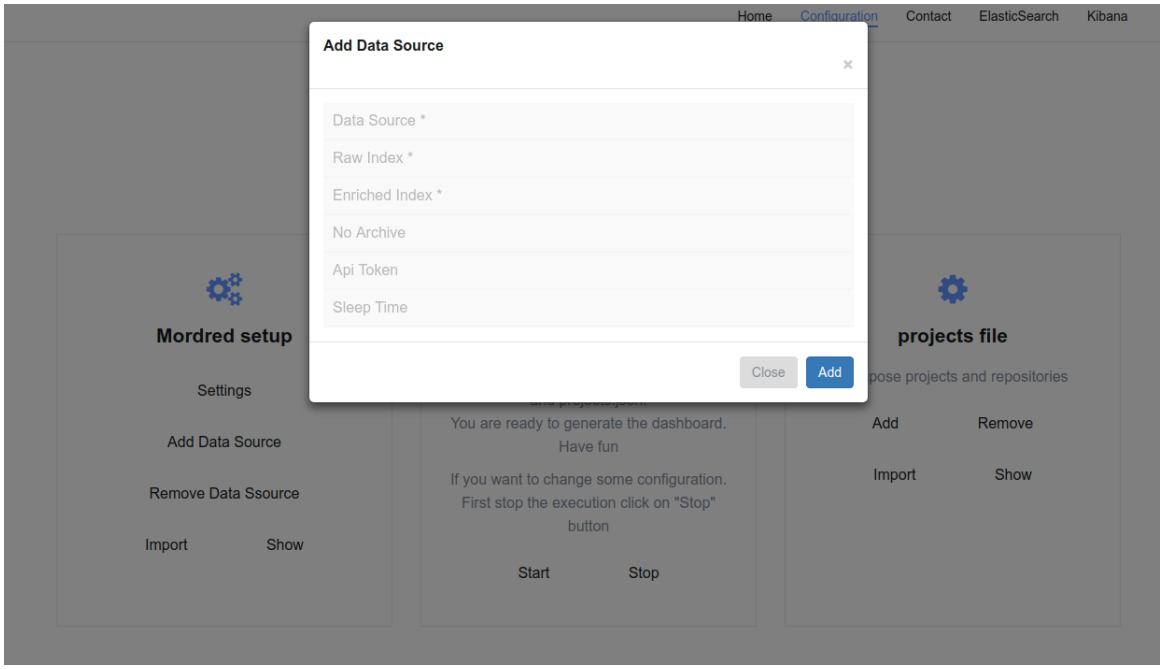

Figure 3.4: Add Data Source

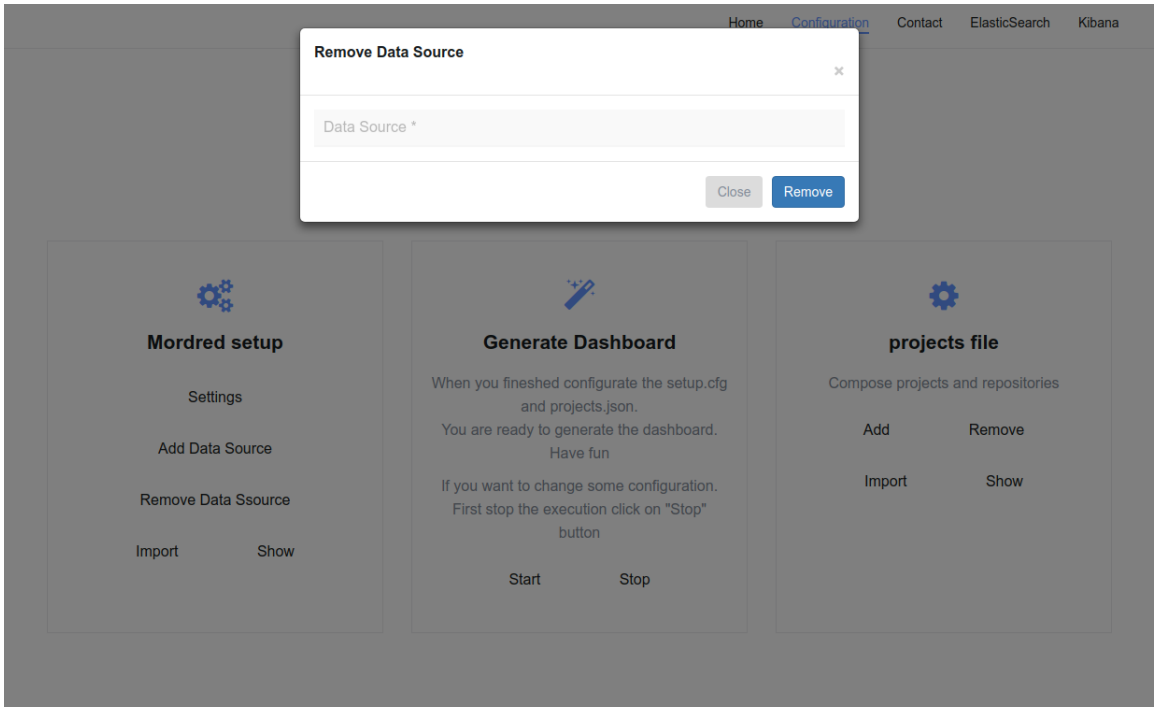

Figure 3.5: Remove Data Source

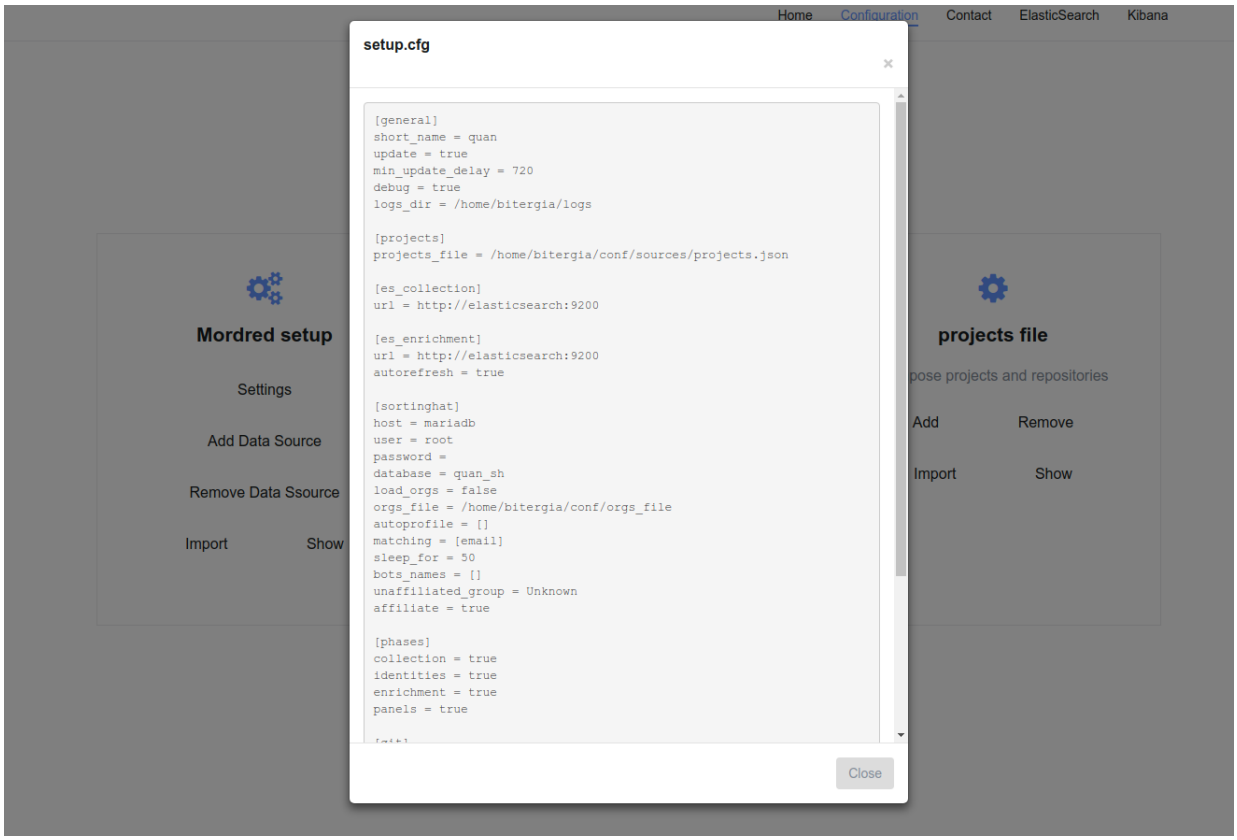

Figure 3.6: Show SirMordred Setup

Chapter 4

Resultado final

El resultado es una aplicación en la que cualquiera puede configurar los ficheros y crear un dashboard en menos de 10 minutos (dependiendo del tamaño del proyecto, ya que la extracción y el enriquecimiento de los datos es directamente proporcional).

Los datos que se han usado para este proyecto han sido sacados del proyecto CHAOSS y GrimoireLab.

Todo el proceso de extracción, enriquecimiento, escritura en ElasticSearch y las visualizaciones en Kibana han sido desarrollados principalmente por Bitergia llamado GrimoireLab.

Esta aplicación puede ahorrar mucho dinero a las empresas y/u organizaciones que quieran saber como están evolucionando sus proyecto, para tomar decisiones determinantes, para mejorarlo o decidir abandonarlo porque no tenga futuro.

Una vez terminada la aplicación vamos a ver como ha quedado.

Para la página de inicio (Figure 4.1, Figure 4.2) vemos una interfaz muy simple, pero elegante y con de fácil creación usando Bootstrap. La configuración general de SirMordred (Figure 3.3) es bastante sencilla con esta interfaz, pero lo es aún más la configuración del fichero de proyecto (Figure 4.3).

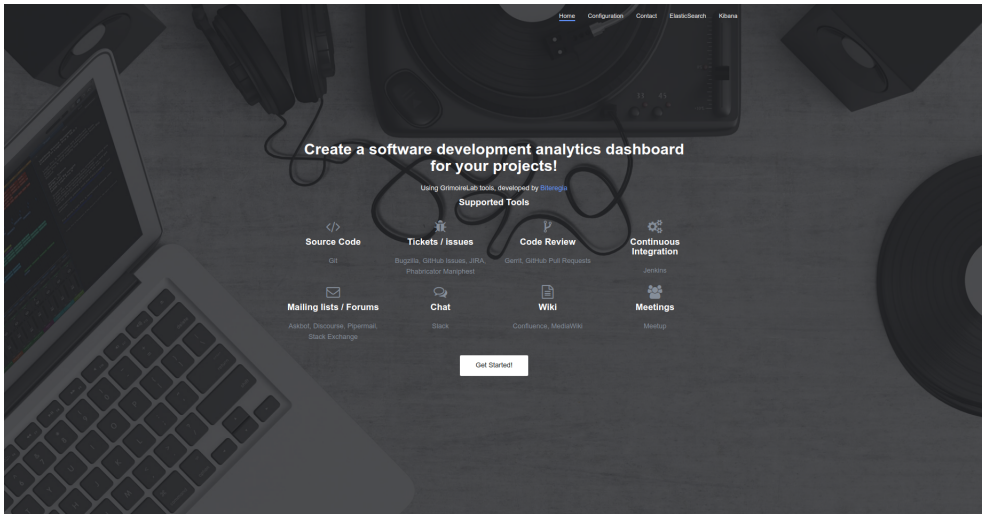

Figure 4.1: Overview

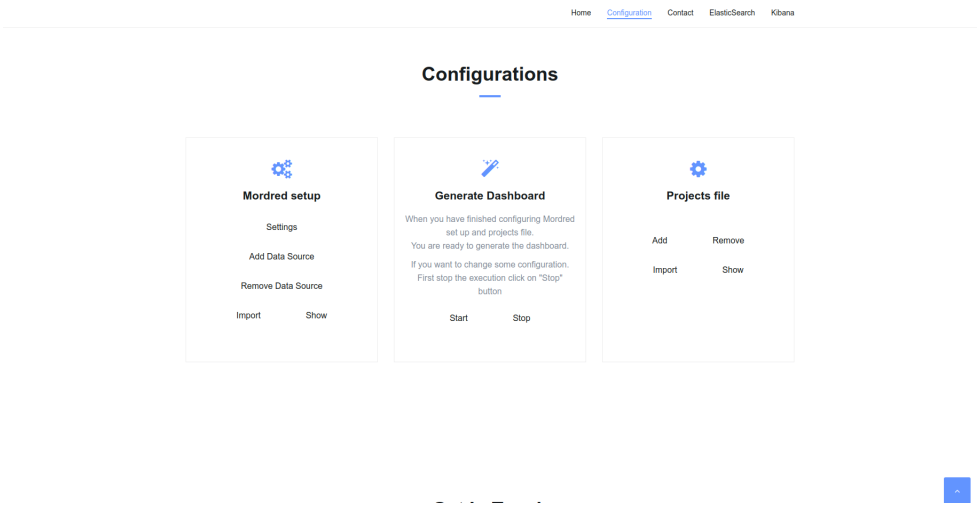

Figure 4.2: Configuration

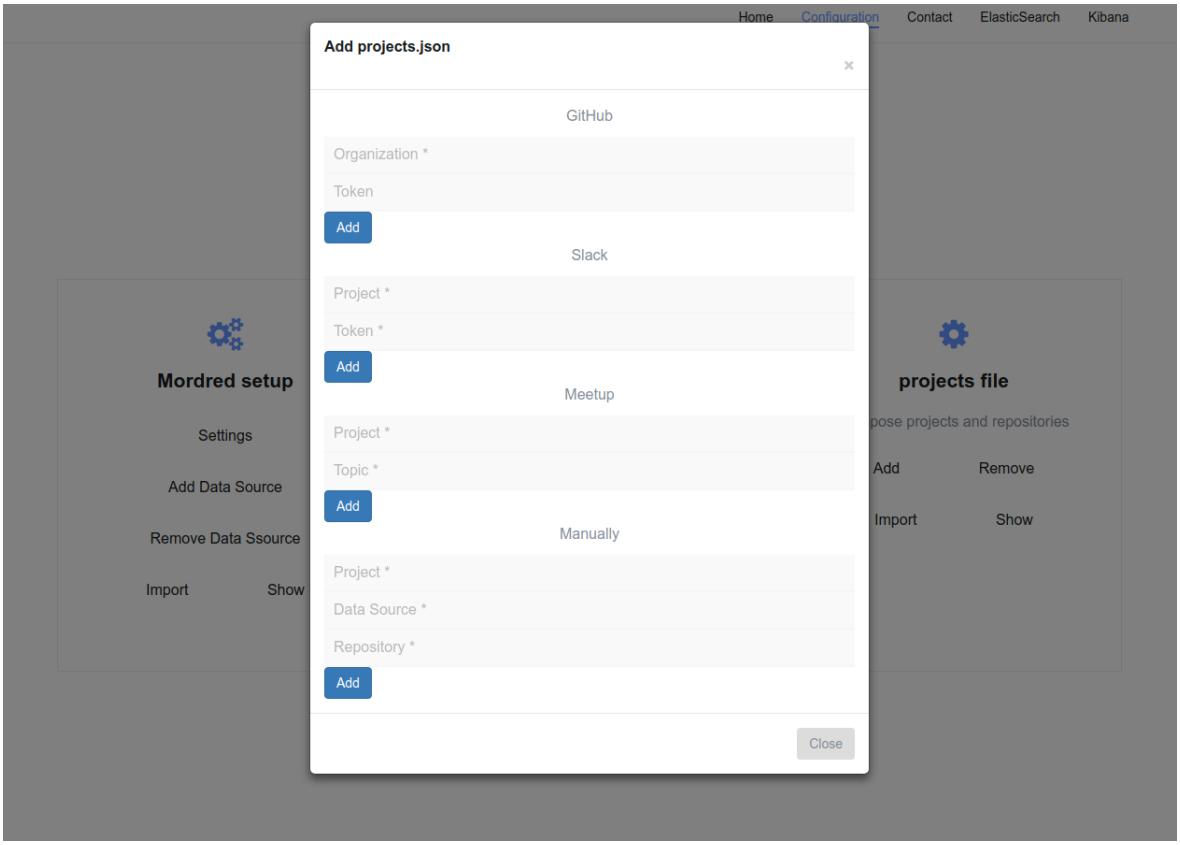

Figure 4.3: Add projects.json

4.1 Demostración

Lo más importante ahora es saber si realmente es sencillo desplegar un dashboard desde cero con esta aplicación.

En primer lugar, debemos tener un Elasticsearch, un Kibana y un MariaDB. Lo vamos a crear en contenedores docker usando docker-compose.yml que es bastante sencillo. Se ha puesto en dos docker-compose.yml diferente para separar Elastic y MariaDB, pero puedes ponerlos todos juntos si lo prefieres.

```
    $ mkdir elastic
    $ cd elastic
    $ vim docker-compose.yml
elasticsearch:
  restart: on-failure:5
  image: elasticsearch:5.1.1
  command: elasticsearch -Enetwork.bind_host=0.0.0.0 \
           -Ehttp.max_content_length=2000mb
  environment:
    - ES_JAVA_OPTS=-Xms2g -Xmx2g
  expose:
    - "9200"
  ports:
    - "9200:9200"
  log_driver: "json-file"
  log_opt:
    max-size: "100m"
    max-file: "3"

kibana:
  restart: on-failure:5
  image: kibana:5.1.1
  environment:
    - NODE_OPTIONS=--max-old-space-size=1000
  links:
    - elasticsearch
  ports:
    - "5601:5601"
  log_driver: "json-file"
```

```
  log_opt:
      max-size: "100m"
      max-file: "3"

    $ cd ..
    $ mkdir mariadb
    $ cd mariadb
    $ vim docker-compose.yml
mariadb:
  restart: on-failure:5
  image: mariadb:10.0
  expose:
    - "3306"
  environment:
    - MYSQL_ROOT_PASSWORD=
    - MYSQL_ALLOW_EMPTY_PASSWORD=yes
  command: --wait_timeout=2592000 --interactive_timeout=2592000 \
           --max_connections=300
  log_driver: "json-file"
  log_opt:
      max-size: "100m"
      max-file: "3"
```

El momento de crear el contenedor de SirMordred con docker-compose.yml ha llegado. Aquí hay que prestar especial atención a la parte de volúmenes, porque tendremos que crear también la carpeta de 'logs' donde escribe SirMordred toda su actividad y si hay algún error o no. En el 'conf' será donde se creen los ficheros de proyectos y el de configuración.

```
    $ cd ..
    $ mkdir mordred
    $ cd mordred
    $ mkdir logs
    $ vim docker-compose.yml
redis:
  image: redis

mordred:
  restart: on-failure:5
  image: bitergia/mordred:18.05-02
```

```
  volumes:
    - /home/quan/mordred:/home/bitergia/conf
    - /home/quan/mordred/logs:/home/bitergia/logs
  links:
    - redis
  external_links:
    - elasticsearch_elasticsearch_1:elasticsearch
    - mariadb_mariadb_1:mariadb

  log_driver: "json-file"
  log_opt:
      max-size: "100m"
      max-file: "3"
```

Hay que tener en cuenta que el docker-compose.yml de SirMordred está en '/home/quan/mordred/docker-compose.yml' porque este path se lo tendremos que pasar a la aplicación con el fin de que coja este docker-compose.yml. ¿Por qué está hecho así? Para que podamos generar varios dashboards diferentes y con sólo introducir donde está el 'docker-compose.yml' podremos usar la misma aplicación para varios dashboards.

A la hora de empezar a usar la aplicación, lo primero que tenemos que hacer es clonar el repositorio.

```
$ cd ..
$ git clone https://github.com/zhquan/Q-DashMan.git
```

Lanzamos el servidor Django para que podamos empezar a jugar.

```
$ cd Q-DashMan
$ python3 python3 manage.py runserver
Performing system checks...
System check identified no issues (0 silenced).
June 28, 2018 - 17:03:35
Django version 2.0, using settings 'tfm.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CONTROL-C
```

La aplicación ya está en funcionamiento. ¡Comencemos a jugar con ella! Esta es la página principal como vimos ya en la Figure 4.1

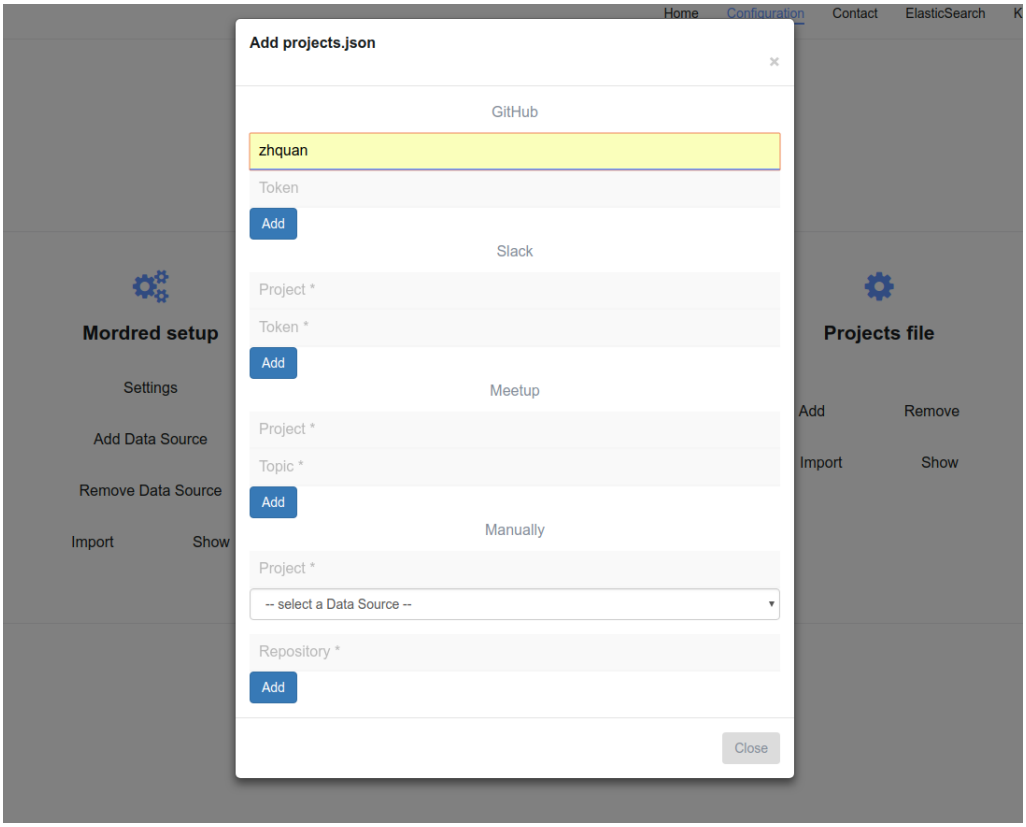

Figure 4.4: Add projects zhquan

El fichero de proyectos es lo primero que hemos de configurar, así que hacemos scroll o click en 'configuration' en la parte superior o en 'Get Started!' en la parte inferior. En la parte de la configuración que se ve como en Figure 4.2, hacemos click en 'Add' de 'projects file' donde nos aparecerá el formulario de la Figure 4.4. En este caso, he añadido mi usuario 'zhquan' en la parte de GitHub.

Para ver que hemos añadido bien hacemos click en 'show', y efectivamente vemos en la Figure 4.5 que ya están añadidos todos los repositorios.

Ahora nos toca configurar el fichero de SirMordred, haremos click en 'Settings' en la parte de 'Mordred setup' y rellenaremos el formulario como se muestra en Figure 4.6 y Figure 4.7.

A continuación, añadimos las dos fuentes de datos que son git y GitHub al fichero de configuración, hacemos click en 'Add Data Sources' y rellenamos el formulario dos veces, uno por cada fuente, como se ve en la Figure 4.8 y Figure 4.9.

Lo recomendable es cercionarse de haber configurado bien el fichero y como se observa en la Figure 4.10 hemos terminado con este fichero también.

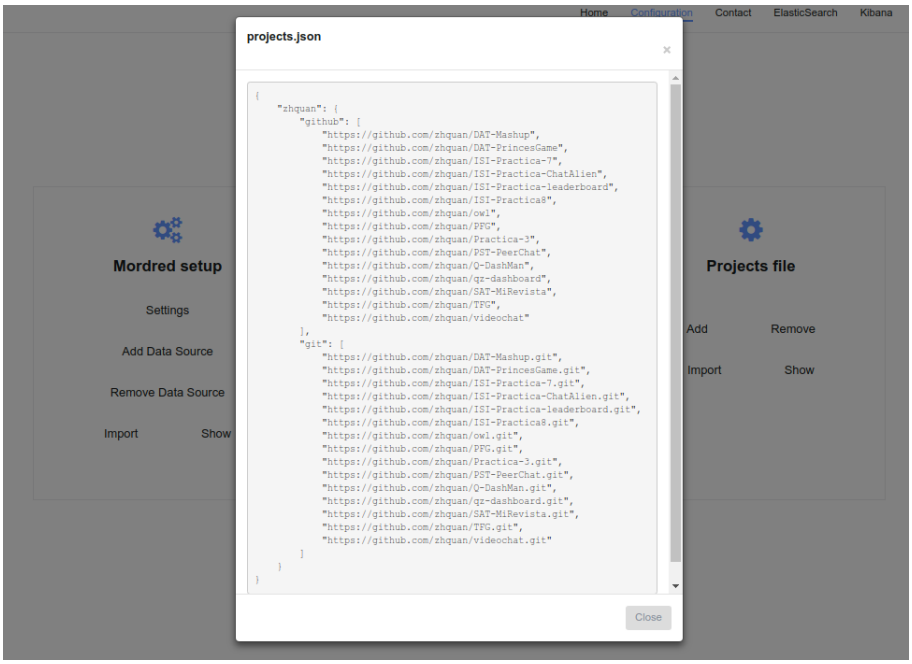

Figure 4.5: Show projects zhquan

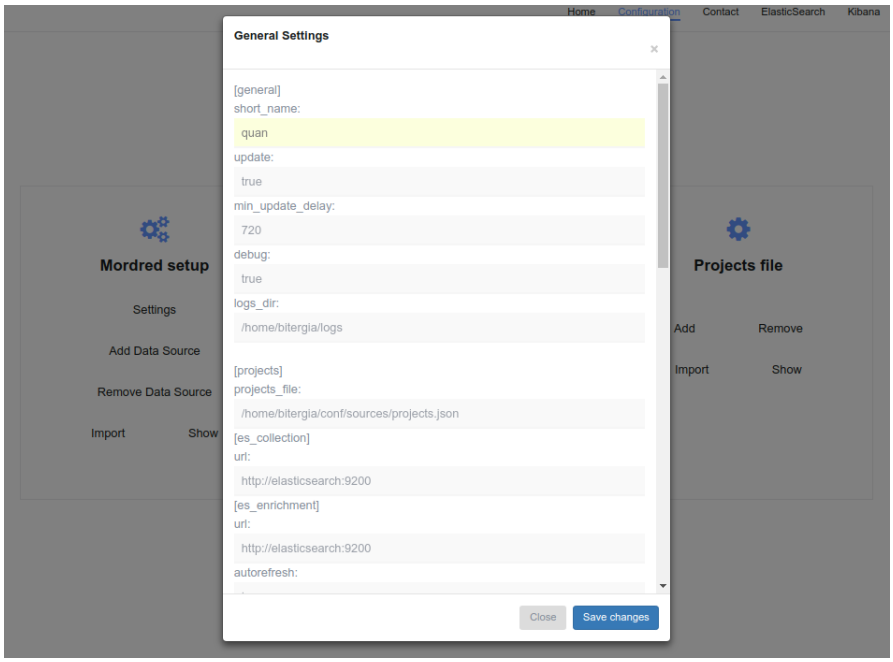

Figure 4.6: Set up settings quan 1

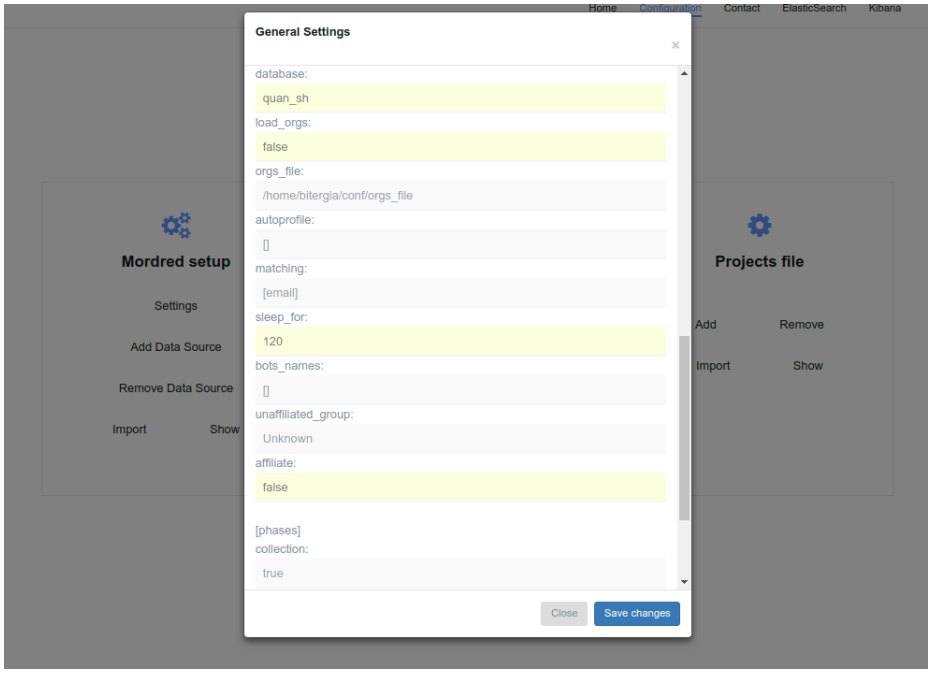

Figure 4.7: Set up settings quan 2

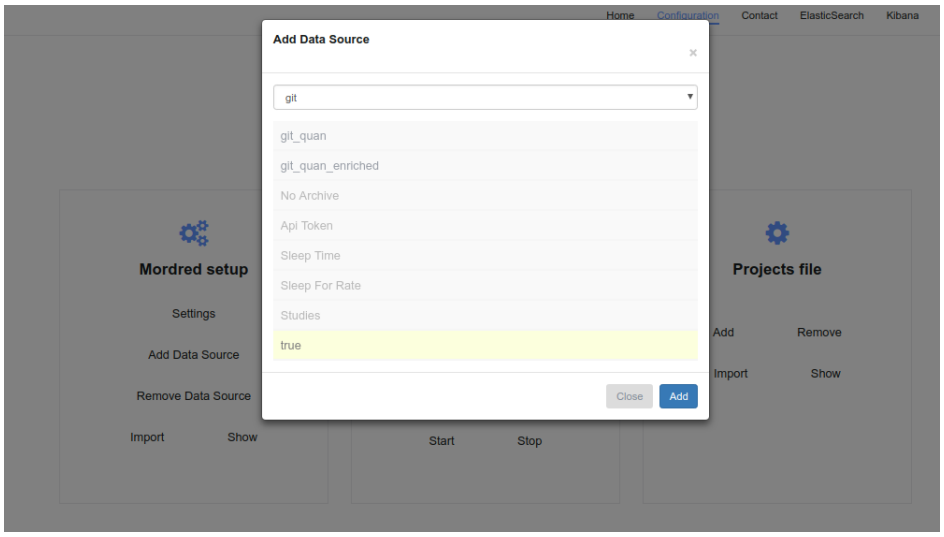

Figure 4.8: Set up data source git

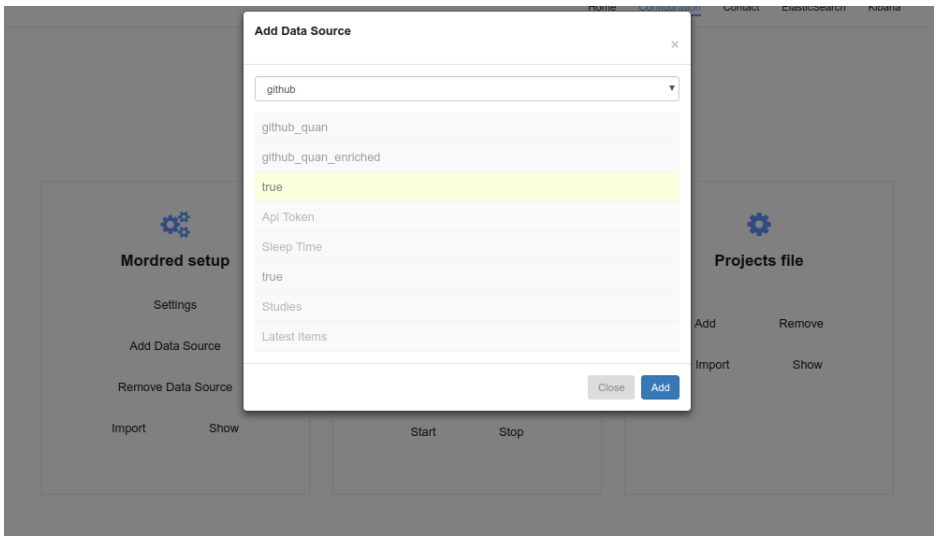

Figure 4.9: Set up data source GitHub

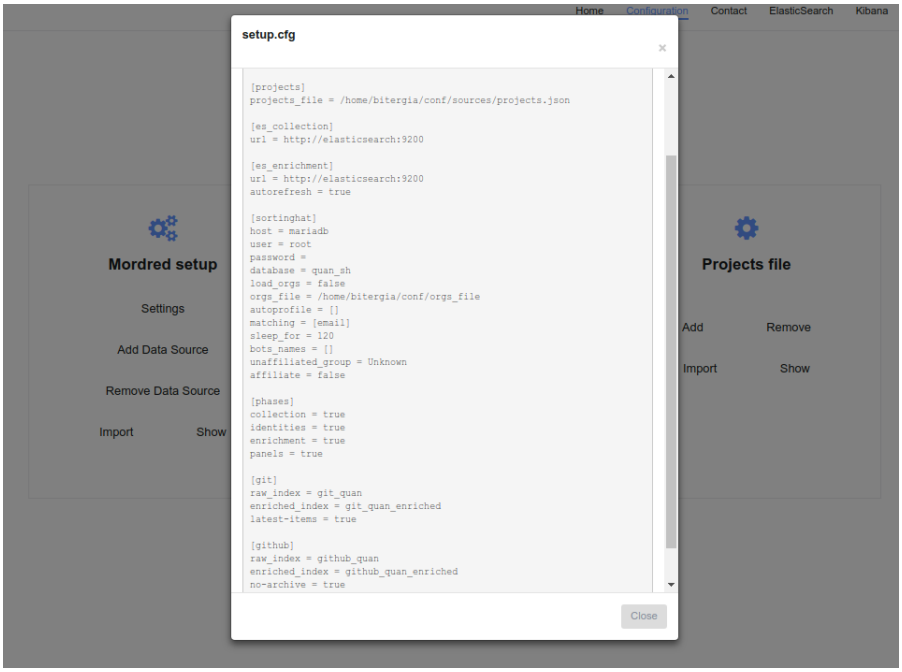

Figure 4.10: Show Mordred set up file

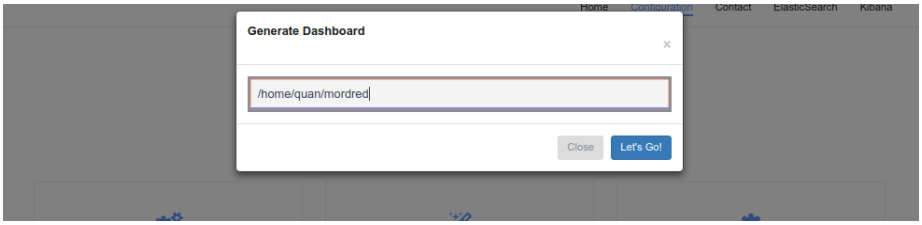

Figure 4.11: Generate dashboard

```
yellow open github_quan_enriched I4Mm8QipSBmzzYFJuzPzXw 5 1   1   0  22.9kb  22.9kb
yellow open git_quan             DSDoJ8UkT2KjuhHnISMzrw 5 1 291   0   5.6mb   5.6mb
yellow open .kibana              soxC6o_XTFCOOAz3H8kr6g 1 1 176   3 249.3kb 249.3kb
yellow open github_quan          CYhgstG_Q7SFuziHPFzBSQ 5 1   1   0  39.7kb  39.7kb
yellow open git_quan_enriched    PnUPsPXFSL6A_gcZTMd4wg 5 1 291 282   1.7mb   1.7mb
yellow open github               83ub4aj6TtmCGRHSEEB7yQ 5 1   1   0   4.6kb   4.6kb
yellow open conf                 a-kaWC6PTLSlD5I_T4Hy2A 5 1   1   0   7.9kb   7.9kb
```

Figure 4.12: Elasticsearch

Por último, sólo nos queda generar el dashboard, así que hacemos click en 'Start' en la parte de 'Generate Dashboard' y escribimos el path de nuestro docker-compose.yml del SirMordred, al igual que en la Figure 4.11

Después de unos minutos, nuestro Kibana ya habrá extraído los datos, los habrá procesado y estarán preparados para ser mostrados, pero si queremos saber como está evolucionando, hacemos click en 'elasticsearch' en la parte superior o directamente a `http://localhost:9200/_cat/indices` para ver si se crean los índices.

Una vez que se hayan creado los índices enriquecidos y que el número de documentos sea igual que los crudos como se ve en la Figure 4.12

Posteriormente, hacemos click en 'kibana' en la parte superior o `http://locahos:5601` para ver los paneles de git y de GitHub. Veamos el panel de git, como se aprecia en la Figure 4.13 y Figure 4.14.

En menos de 10 minutos podemos crear un dahsboard personalizado de una manera muy sencilla e intuitiva, que es lo que se pretendía con esta aplicación.

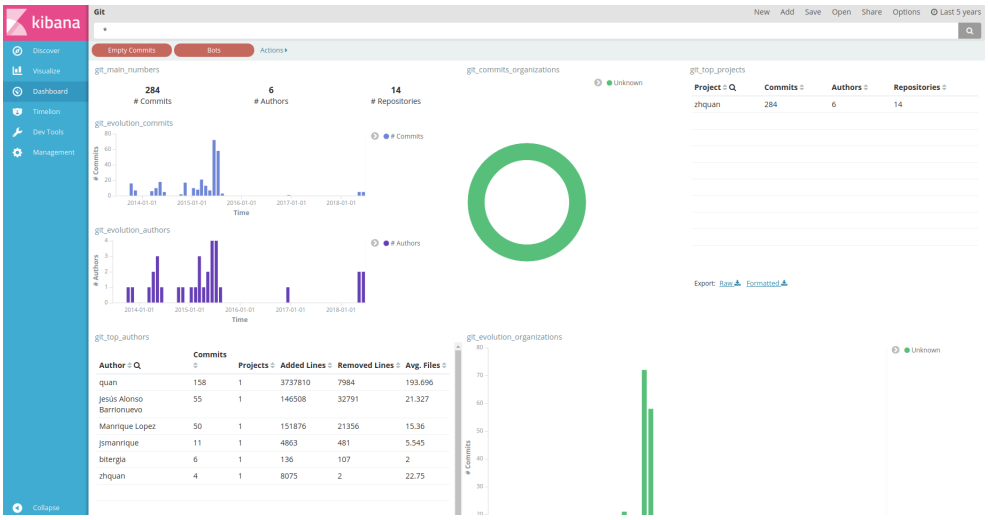

Figure 4.13: Kibana git 1

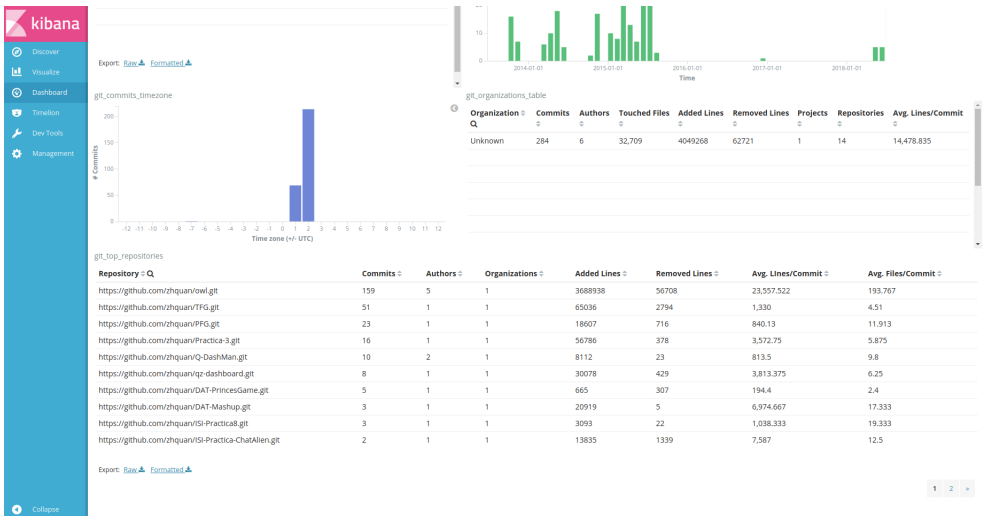

Figure 4.14: Kibana git 2

Chapter 5

Conclusiones

Este proyecto no hubiera sido posible sin los grandes avances realizados por la empresa Bitergia y sin los conocimientos que he adquirido gracias a encontrarme en su nucleo al ser parte de la misma.

Q-DashMan es una aplicación web donde puedes crear tu propio dashboard personalizado con un análisis de los datos en profundidad de las herramientas de desarrollo de software como GitHub, MediaWiki, Jira, Confluence, Slack, Meetup, etc. Para este proceso tan solo necesitamos unos pocos y simples pasos, como hemos visto en el capítulo ”4.1 Demostración” donde se detallan paso a paso desde cómo hacer la instalación de la aplicación hasta cómo obtener nuestro dashboard. Los datos se van actualizando según la actividad que vayan teniendo, porque es un proceso incremental que nunca para de extraer datos de las fuentes. Aquello que vemos, es prácticamente en tiempo real.

Esta aplicación es útil para ver como se están evolucionando sus proyectos o de cualquier otro proyecto open source. En un mundo en el que los datos son muy importantes y valiosos, ya que con unos pequeños cálculos somos capaces de saber cuáles son los gustos de las personas, en que gasta su tiempo libre, en que trabajan..., podemos usar esa información para adelantarnos a la competencia o tomar decisiones importantes para el futuro del proyecto o más importante aún, de la propia empresa.
Pueden saber como están evolucionando sus proyectos, las personas que están contribuyendo y quienes son los principales contribuidores para poder tomar las mejores decisiones.

La aplicación está alojada en GitHub, una plataforma de control de versiones que es usado por millones de usuarios.

El proyecto se desarrolló exclusivamente usando software libre y de hecho esta aplicación también es software libre con licencia GNU General Public License (GPL), version 3.0.

5.1 Lecciones aprendidas

Esta experiencia ha sido muy enriquecedora, por lo que procedemos a ensalzar algunas de las lecciones aprendidas:

- Comprender de la variedad de medios de colaboración existentes en el desarrollo del software libre.
- Informar sobre fallos detectados en un programa o aplicación, siendo consciente de la importancia para el bien común que supone el hacerlo.
- Proponer mejoras a los productos actuales que existen en las empresas y de saber mejorarlos.
- Buscar información en diversas fuentes como foros, libros, etc.
- Escribir un código más limpio y legible.
- Autogestionar el tiempo y el ritmo para que el proyecto este terminado en una fecha concreta.
- Mejorar en el uso de la herramienta GitHub donde se encuentra alojado este proyecto y otros trabajos realizados con anterioridad.
- Aprender a manejar LaTeX, el sistema de procesamiento de documentos empleado en la realización de esta memoria y de otros tantos documentos en el futuro, con total seguridad.

5.2 Conocimientos aplicados

Los conocimientos que se han aplicado para lograr con éxito este proyecto han sido exclusivamente la experiencia que he obtenido trabajando en Bitergia. Como parte del equipo de operaciones teníamos muchas cosas que mejorar, pero estaba especialmente interesado en como

estábamos desplegando los dashboards para los clientes, el tiempo que gastábamos para configurar todos los ficheros y las versiones de las imágenes de ElasticSearch, MariaDB, Kibana y SirMordred. Debido a los problemas que teníamos a la hora de desplegar un dasshboard y de haber estado investigando cual era la mejor manera de hacerlo, pero que a su vez fuera sencillo para el operador o cualquier otra persona de la empresa, pude desarrollar esta aplicación adecuadamente.

Tanto la aplicación como sus herramientas se han desarrollado casi en su totalidad con el lenguaje de programción Python la version 3.5. Aunque también se ha usado JavaScript, HTML y CSS para el front-end y el framework Django en la parte del servidor.

Por otro lado, se ha usado ElasticSearch como la base de datos principal, donde se guarda toda la información en los índices crudos y enriquecidos y MariaDB para los datos de afiliación como el género, país y la organización donde trabaja. Por último, Kibana para la visualización de los datos, ya que puede crear gráficas personalizadas en el interfaz de usuario directamente.

5.3 Trabajos futuros

Una vez finalizado el proyecto, se plantean varios campos de trabajo que supondrían una extensión de su funcionalidad y contribuirían a consolidarlo.

La configuración de Sortinghat se puede añadir para que se pueda entrar directamente a la base de datos y poder afiliar a mano, lo cual no es posible en la actualidad y es una funcionalidad muy útil para así tener más control acerca de la afiliación siendo un tema muy peliagudo.

Si añadimos KingArthur a la aplicación hacemos que sea mucho más rápida y eficiente, ya que esta herramienta está en su última fase de pruebas y en poco tiempo estará integrado en el producto.

Otra vía de mejora de este proyecto sería generar de forma automática informes usando Manuscripts para completar más esta aplicación y sobretodo para que se entienda mucho mejor lo qué se está mostrando y cómo van evolucionando cuatrimestralmente.

Todos estos nuevos objetivos (y los que pueda plantear una tercera persona), podrán implementarse para potenciar la herramienta, que quedará amparada por una licencia de software libre, específicamente el GPL v3.0

Bibliography

[1] Python.

https://www.python.org/

https://docs.python.org/3/

https://es.wikipedia.org/wiki/Python

[2] Django.

https://www.djangoproject.com/

[3] JavaScript.

http://es.wikipedia.org/wiki/JavaScript

http://www.w3schools.com/js/default.asp

http://librosweb.es/libro/javascript/

[4] HTML.

http://es.wikipedia.org/wiki/html

http://www.w3schools.com/html/default.asp

[5] CSS.

http://www.w3schools.com/css/default.asp

http://es.wikipedia.org/wiki/css

http://librosweb.es/libro/css/

[6] JQuery.

https://jquery.com/

http://www.w3schools.com/jquery/default.asp

http://librosweb.es/libro/fundamentos_jquery/

[7] Bootstrap.

http://librosweb.es/libro/bootstrap_3/

http://getbootstrap.com/

[8] Elastic.

https://www.elastic.co/

https://es.wikipedia.org/wiki/Elasticsearch

[9] Slack.

https://slack.com/

[10] Phabricator.

https://www.phacility.com/

https://secure.phabricator.com/

[11] Jira.

https://es.atlassian.com/software/jira/

[12] Jenkins.

https://jenkins.io/

[13] Askbot.

https://askbot.com/

https://askbot.org/

[14] Discourse.

https://www.discourse.org/

[15] StackExchange.

https://stackexchange.com/

[16] Confluence.

https://es.atlassian.com/software/confluence/

[17] MediaWiki.

https://www.mediawiki.org/wiki/MediaWiki/

[18] Meetup.

`https://www.meetup.com/`

[19] Git.

`https://git-scm.com//`

[20] GitLab.

`http://gitlab.com/`

[21] GitHub.

`https://github.com`

[22] Bitergia.

`http://bitergia.com/`

[23] Perceval.

`https://github.com/chaoss/grimoirelab-perceval/`

`http://perceval.readthedocs.io`

[24] Sortinghat.

`https://github.com/chaoss/grimoirelab-sortinghat/`

[25] Sigils.

`https://github.com/chaoss/grimoirelab-sigils`

[26] GELK.

`https://github.com/chaoss/grimoirelab-elk/`

[27] SirMordred.

`https://github.com/chaoss/grimoirelab-sirmordred/`

[28] KingArthur.

`https://github.com/chaoss/grimoirelab-kingarthur/`

[29] Manuscripts.

`https://github.com/chaoss/grimoirelab-manuscripts/`