



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

GRADO EN INGENIERÍA EN SISTEMAS DE TELECOMUNICACIÓN

TRABAJO FIN DE GRADO

CREACIÓN DE MÓDULOS DE VISUALIZACIÓN DE DATOS PARA KIBANA

Autor : Momchil Stoyanov Momchilov

Tutor : Dr. Jesús María González Barahona

Curso académico 2016/2017

Proyecto Fin de Grado

CREACIÓN DE MÓDULOS DE VISUALIZACIÓN DE DATOS PARA KIBANA

Autor : Momchil Stoyanov Momchilov

Tutor : Dr. Jesús María González Barahona

La defensa del presente Proyecto Fin de Carrera se realizó el día de
de 2017, siendo calificada por el siguiente tribunal:

Presidente:

Secretario:

Vocal:

y habiendo obtenido la siguiente calificación:

Calificación:

Fuenlabrada, a de de 2017

*Dedicado a
mi familia.*

Agradecimientos

Quiero dar las gracias, en primer lugar, a mis padres ya que sin su esfuerzo diario nada de esto sería posible. Gracias por todo el apoyo que me han dado y por haberme enseñado a luchar por mis sueños y no rendirme jamás. También me gustaría agradecer a mi tutor Jesús M. González Barahona la oportunidad de trabajar con él, por haber confiado en mí y por toda su dedicación y apoyo lo largo de todo el proyecto.

A mis amigos de toda la vida y a aquellos que he ido conociendo a lo largo de los años en la universidad, gracias por haber compartido numerosas experiencias a mi lado, tanto buenas como malas, y por todo vuestro apoyo. Por último dar las gracias a mis compañeros de la empresa en la que empecé mi aventura en el mundo laboral que desde el primer día me he sentido integrado y me han tratado como a uno más del equipo.

Especial mención a todos aquellos que comparten su conocimiento a través de la red en numerosos foros y plataformas web y que me han ayudado a superar todos los desafíos que ha planteado el desarrollo de este proyecto. Muchas gracias héroes anónimos.

Resumen

La finalidad de este proyecto es la creación de un módulo de visualización de datos para Kibana 5 con la intención de ampliar la variedad de representaciones gráficas ofrecidas a los usuarios e incluir nuevas funcionalidades. El plugin desarrollado utiliza el software de código abierto (OSS) Kibana que sirve para visualizar y explorar la información almacenada en la base de datos NoSQL Elasticsearch. Para seleccionar la biblioteca gráfica de JavaScript se ha hecho un estudio de las principales alternativas disponibles actualmente y posteriormente se ha llegado a la conclusión de que C3.js es la más adecuada para este proyecto. La funcionalidad más importante es la posibilidad de integrar varias representaciones gráficas en un cuadro de mando.

La idea que ha impulsado la realización del proyecto ha sido el análisis y la representación de grandes volúmenes de datos para mejorar la toma de decisiones. Para llevar a cabo la implementación del plugin se ha utilizado el framework AngularJS para escribir el código JavaScript, HTML y CSS. El proyecto cuenta con su propia página web y se aloja en GitHub. Diversos usuarios y desarrolladores han probado su funcionamiento y han contribuido al desarrollo de la aplicación informando sobre errores y aportando ideas.

El mayor logro es que los creadores de Kibana y Elasticsearch han publicado este plugin en su página oficial lo que ha permitido que la aplicación gane popularidad y usabilidad entre los usuarios.

Summary

The main aim of this project is the creation of a data visualization module for Kibana 5 with the intention of expanding the variety of graphic representations offered to the users and include new features. This plugin uses Kibana, an open source analytics and visualization platform, and a search engine called Elasticsearch. To select the JavaScript library has made a study of the main alternatives currently available and has subsequently come to the conclusion that C3.js is the most suitable for the aim of this project. The most important functionality is the possibility of integrating several representations into a dashboard.

The idea behind the project has been the analysis and representation of large volumes of data to improve decision making. This plugin has been developed with AngularJS in order to write the JavaScript, HTML and CSS code. The project has its own website and is hosted on GitHub. Many users and developers have tested this software and some of them have contributed to the development of the application reporting errors and providing fresh ideas.

The biggest achievement is that the creators of Kibana and Elasticsearch have published this plugin in their official page which has allowed the application to gain popularity and usability among a lot of users.

Acrónimos

API: Application Programming Interface

CORS: Cross-origin Resource Sharing

CPU: Central Processing Unit

CSS: Cascading Style Sheets

DOM: Document Object Model

ECMA: European Computer Manufacturers Association

HTML: HyperText Markup Language

HTTP: Hypertext Transfer Protocol

HTTPS: Hypertext Transfer Protocol Secure

IoT: Internet of Things

JS: JavaScript

JSON: JavaScript Object Notation

KPI: Key Performance Indicator

NPM: Node Package Manager

OSS: Open Source Software

SDK: Software Development Kit

SPA: Single Page Application

SQL: Structured Query Language

REST: Representational State Transfer

W3C: World Wide Web Consortium

XML: eXtensible Markup

XSLT: EXtensible Stylesheet Language

Índice general

1. Introducción	1
1.1. Contexto	1
1.2. Motivación personal	3
1.3. Objetivos del proyecto	3
1.4. Estructura de la memoria	5
1.5. Disponibilidad del Software	6
2. Estado del arte y tecnologías	7
2.1. Elasticsearch	7
2.2. Kibana	9
2.3. JavaScript	11
2.4. AngularJS	12
2.5. D3.js	14
2.6. C3.js	14
2.7. JSON	15
2.8. NodeJS y NPM	15
2.9. Elasticdump	17
2.10. HTML	17
2.11. Bootstrap	18
2.12. CSS	19
2.13. LESS	19
2.14. Git	20
2.15. GitHub	21
2.16. Metodología SCRUM	21

2.17. Herramientas para representar datos	23
3. Diseño e implementación	27
3.1. Modelo de desarrollo	27
3.2. Iteración 0. Estudio previo	29
3.2.1. Objetivos	29
3.2.2. Aprendizaje de JavaScript y AngularJS	29
3.2.3. Desarrollo primera aplicación	30
3.3. Iteración 1. Elasticsearch y bibliotecas gráficas	32
3.3.1. Objetivos	32
3.3.2. Elasticsearch	32
3.3.3. Integración de Elasticsearch con AngularJS	33
3.3.4. Bibliotecas gráficas	35
3.4. Iteración 2. Dashboard propio y Kibana	36
3.4.1. Objetivos	36
3.4.2. Desarrollar mi propio dashboard	36
3.4.3. Kibana	37
3.5. Iteración 3. Construcción de widgets en Kibana	38
3.5.1. Objetivos	38
3.5.2. El primer widget	39
3.5.3. Integración de C3.js en Kibana 5	40
3.6. Iteración 4. Implementación del plugin	41
3.6.1. Objetivos	41
3.6.2. Nuevas visualizaciones	41
3.6.3. Especificaciones añadidas	43
3.6.4. Publicación del plugin y sitio web	44
4. Resultados	45
4.1. Arquitectura del plugin	45
4.2. Funcionamiento del widget	47

5. Conclusiones	53
5.1. Aplicación de lo aprendido	54
5.2. Lecciones aprendidas	55
5.3. Trabajos futuros	55
A. Código de la aplicación	57
A.1. index.js	57
A.2. package.json	57
A.3. c3_charts_vis.html	58
A.4. c3_charts_vis.js	58
A.5. c3_charts_vis.less	61
A.6. c3_charts_vis_controller.js	61
A.7. c3_charts_vis_params.html	71
B. Licencia del proyecto	79
Bibliografía	81

Índice de figuras

1.1. Ejemplo de gráfico Microsoft Office	2
2.1. Glosario sistema distribuido	8
2.2. Ejemplo de agregaciones	10
2.3. Diseño Kibana 5	10
2.4. Gráficos D3	14
2.5. Variable LESS	20
2.6. Ejemplo contribuciones GitHub	21
2.7. Esquema del modelo Scrum	22
2.8. Bitergia	23
2.9. Cuadro de mando Cyfe	24
2.10. Visualizaciones en Grafana	25
3.1. Conjunto de herramientas para desarrollo web	30
3.2. Añadir y filtrar contactos	31
3.3. Lista de contactos	31
3.4. Cluster state and indexes	33
3.5. Tabla de datos extraídos de Elasticsearch	34
3.6. Visualizaciones con datos de Elasticsearch	35
3.7. Representación gráfica con Superset	37
3.8. Dashboard en Kibana 4	38
3.9. El reloj digital en Kibana 4	39
3.10. Medidor radial en Kibana 5	40
3.11. Ajuste anchura del gráfico de barras	42

3.12. Dashboard Kibana 5	44
4.1. Diagrama de archivos	46
4.2. Lista de visualizaciones	47
4.3. Apariencia de la aplicación	48
4.4. Panel de opciones	49
4.5. Cuadro de descripción	50
4.6. Opción de enfoque	51
4.7. Integración de visualizaciones en un cuadro de mando	52

Capítulo 1

Introducción

Este proyecto trata sobre la representación de datos de forma gráfica basándose en el *framework*, de software libre, Kibana y se apoya en una biblioteca de visualización JavaScript denominada C3.js. El objetivo principal es tratar de extender dicho *framework* y ampliar la variedad de visualizaciones ofrecidas al usuario, incluyendo algunas funcionalidades que no existen en la actualidad.

A continuación, para una mejor comprensión del proyecto, se explicarán el contexto del mismo, las motivaciones que me han llevado a realizarlo y la estructura de la memoria.

1.1. Contexto

A día de hoy, en el mundo empresarial se requiere, cada vez más, de tecnología y herramientas punteras para satisfacer las necesidades de los clientes. En un mundo globalizado en el que el *Big Data* se hace más presente y el análisis de los datos se vuelve imprescindible para tomar decisiones [3], los ingenieros se ven obligados a proporcionar soluciones tecnológicas sencillas, funcionales y con un diseño atractivo.

En primer lugar, con la llegada de los sistemas operativos modernos, aparecen las primeras visualizaciones sencillas como por ejemplo las aplicaciones ofrecidas por el paquete *Microsoft Office* que permiten a los usuarios crear gráficas en sus informes y/o proyectos y manejar datos de forma sencilla. Posteriormente surgen los primeros *dashboards* como una representación

gráfica de los principales indicadores *KPI* (key performance indicator, o indicador clave de rendimiento) que intervienen en la consecución de los objetivos de negocio, y que están orientados a la toma de decisiones para optimizar la estrategia de la empresa. El principal problema de dichas visualizaciones desarrolladas hasta la fecha es su apariencia estática que imposibilita la interacción del usuario con los datos.

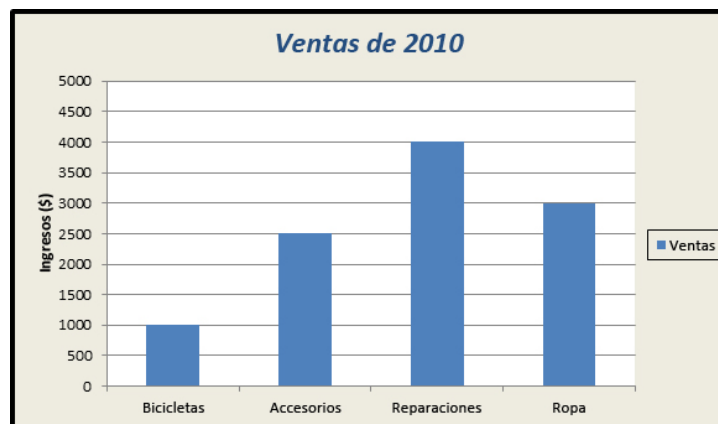


Figura 1.1: Ejemplo de gráfico Microsoft Office

Posteriormente, con la llegada de la Web 2.0, el usuario empieza a cobrar protagonismo gracias al desarrollo de tecnologías como AJAX, CSS, RSS, JSON y XML que han favorecido la popularización de los *dashboards* y la creación de *frameworks* completos debido al hecho de que pueden funcionar en un navegador permitiendo cierta personalización y configuración por parte del usuario. Por otra parte, los servicios de *hosting cloud* (alojamiento en servidores virtuales) proporcionan acceso a datos desde cualquier máquina con conexión a Internet eliminando de esta forma las barreras del almacenamiento local. En los siguientes capítulos se mencionarán algunos servicios disponibles actualmente en la red que permiten la visualización y representación de datos.

Y éste es precisamente el objetivo del proyecto, extender un *dashboard* ya existente denominado Kibana para ampliar las visualizaciones ofrecidas a los usuarios gracias a la biblioteca de visualización JavaScript, C3.js. De esta forma se pretende combinar dos aspectos clave en la actualidad: tratamiento de grandes volúmenes de datos (*Big Data*) para permitir al usuario (o incluso cliente) tener una mejor visión y poder sacar conclusiones rápida e intuitivamente. Y por otro lado que sea un servicio web que admita diferentes navegadores y que permita cierta libertad a la hora de utilizar la aplicación.

1.2. Motivación personal

Desde siempre me he considerado una persona inquieta a la que le gusta interesarse por las innovaciones y novedades tecnológicas. Fue una de las razones que me ayudaron a elegir las telecomunicaciones como mi futuro profesional. Lógicamente la primera parte de la carrera se basa en conceptos teóricos, fundamentos físicos y matemáticos. Sin embargo he tenido la gran suerte de tener asignaturas prácticas, impartidas por profesores con vocación de enseñar, en las que uno aprende el funcionamiento real de las cosas que se utilizan diariamente como por ejemplo el mundo digital y la programación, que me parecen muy interesantes.

Una asignatura especial para mí fue “*Servicios y Aplicaciones en Redes de Ordenadores*” que despertó mi interés por la programación de aplicaciones web. Fue entonces cuando conocí a Jesús y me comentó que está muy interesado en los *dashboards* y en la representación gráfica de datos en general. Al enseñarme un ejemplo y contarme un poco sobre el tema me quedé fascinado. Posteriormente acordamos realizar mi TFG de manera conjunta. Estoy encantado de haber podido participar en un proyecto así, un proyecto real que surge de una idea y va tomando forma a medida que avanza el tiempo.

Actualmente en Kibana, *framework* que utiliza mi *plugin* (aplicación), vienen predefinidos varios tipos de gráficos por defecto como por ejemplo barras, líneas o tartas. Sin embargo dichos *widgets* (complementos) tienen ciertas limitaciones como por ejemplo la imposibilidad de mezclar diferentes tipos de visualizaciones en el mismo *canvas*. Otro inconveniente es la inexistencia de un *tooltip* (descripción emergente o globo de ayuda) en el que se muestre información acerca de todos los datos representados en un punto concreto de la gráfica. Este proyecto trata de proporcionar una solución a dichas deficiencias e integrar nuevas funcionalidades utilizando la biblioteca gráfica C3.js [7].

1.3. Objetivos del proyecto

El objetivo principal del proyecto es elaborar un *widget* que sea compatible con Kibana 5 y permita ampliar las visualizaciones ofrecidas por el *framework*. Una de las metas es que el *plugin* debe mantener la interfaz de Kibana para que los usuarios no noten un impacto significativo

respecto a los *widgets* existentes.

Por otro lado, se tratarán de introducir nuevas características y funcionalidades que puedan mejorar el resultado final de las representaciones (zoom, etiquetado de los ejes, combinar diferentes tipos de visualizaciones, personalizar nombres y colores, descartar o recargar datos en tiempo real, *tooltip*, animaciones, enfocar determinados datos con el ratón, modificar la posición de la leyenda, etc.).

Para llevar a cabo el objetivo principal, se han cumplido una serie de objetivos secundarios que se describen a continuación:

- Familiarizarse con las tecnologías que se utilizan a lo largo del proyecto y que se describen en el siguiente capítulo.
- Aprender el lenguaje de programación JavaScript.
- Realizar un estudio preliminar del funcionamiento de Kibana.
- Entender el funcionamiento del motor de búsqueda NoSQL y respetar la REST API para enviar consultas (*queries*) y recibir datos de Elasticsearch.
- Representar los datos obtenidos de Elasticsearch utilizando la biblioteca gráfica C3.js, basada en D3.js, en diferentes tipos de visualizaciones.
- Utilizar el *framework* AngularJS para escribir el código necesario de JavaScript y HTML5.
- Mantener la estética de la interfaz de usuario lo más sencilla e intuitiva posible.
- Corregir la estética de los gráficos de barras cuando se representan grandes volúmenes de datos categóricos.
- Integrar las diferentes representaciones gráficas en un *dashboard* para potenciar la visión global de los datos. Hacer que las visualizaciones sean responsivas y se actualicen al realizar un cambio de dimensión o rango de representación.
- Verificar el correcto funcionamiento del *plugin* utilizando una base de datos que contiene más de 10 mil entradas.

- Alcanzar un mayor grado de conocimiento y aplicar lo aprendido durante los últimos cursos en una herramienta con potencial real de ser utilizada por otros usuarios o empresas.
- Otorgar visibilidad de la aplicación entre la comunidad de usuarios de Kibana en GitHub y proporcionar soporte a los usuarios que lo requieran.
- Crear una página web para el proyecto¹ alojada en GitHub Pages.
- Configurar una máquina virtual para cargar tanto Kibana como el *plugin* desarrollado para poder utilizar la herramienta desde cualquier dispositivo con conexión a Internet.
- Depuración del código para que sea lo más limpio y sencillo posible y optimización de los recursos software.

1.4. Estructura de la memoria

Para facilitar la lectura de esta memoria, se procede a explicar su estructura, detallando qué contenido tendrá cada capítulo:

- Para empezar se presenta un resumen tanto en español como en inglés del proyecto.
- El primer capítulo, denominado *Introducción*, presenta el contexto del proyecto, la motivación que me ha llevado a realizarlo, los objetivos genéricos y los más específicos, la estructura de esta memoria y por último información acerca de la disponibilidad del software.
- En el segundo capítulo, *Estado del Arte y tecnologías*, se explican las tecnologías y metodologías empleadas para elaborar las distintas partes del proyecto y se exponen algunas de las plataformas disponibles hoy en día en la red para la representación gráfica de datos.
- En el tercer capítulo, *Diseño e implementación*, se explica el desarrollo y la implementación del proyecto, definiendo las principales etapas que se han atravesado.
- En el cuarto capítulo, *Resultados*, se muestran los resultados obtenidos al finalizar el proyecto y las soluciones que se han dado a los principales problemas.

¹<https://mstoyano.github.io>

- En el quinto capítulo, *Conclusiones*, se echa la vista hacia atrás y se presenta un resumen de las principales lecciones aprendidas, cuáles han sido los conocimientos aplicados y qué trabajos futuros se pueden seguir desarrollando.

1.5. Disponibilidad del Software

Debido a que este *plugin* se ha dado a conocer dentro de la comunidad de desarrolladores GitHub, los creadores de Elasticsearch y Kibana han tomado la decisión de añadirlo a la guía de usuario de dicho *software*, en la parte conocida como “*Known Plugins*”.

<https://www.elastic.co/guide/en/kibana/current/known-plugins.html>

El proyecto está amparado bajo la licencia de Apache 2.0 y al tratarse de software libre, todos los usuarios que lo deseen pueden tener acceso al código fuente que se encuentra alojado en un repositorio de GitHub:

https://github.com/mstoyano/kbn_c3js_vis

Además el proyecto cuenta con una página web propia en la que el usuario puede encontrar información acerca de la aplicación, imágenes de su funcionamiento y formas de contacto. También se han proporcionado enlaces a la máquina virtual, el código fuente y la versión en PDF de la memoria del proyecto.

<https://mstoyano.github.io>

Capítulo 2

Estado del arte y tecnologías

En este capítulo se presentan las tecnologías que se han utilizado a lo largo del desarrollo de este proyecto. Para ello se expondrá una visión general de las mismas. Posteriormente se presentarán algunas plataformas y herramientas que se utilizan para la representación de datos.

2.1. Elasticsearch



elasticsearch

Elasticsearch es una base de datos NoSQL que ofrece una solución a nuestros problemas de extracción de datos. Esta herramienta fue creada por Shay Banon en el año 2010 y dos años más tarde se funda la empresa Elastic [4] que ofrece diversos servicios.

Elasticsearch se compone de dos capas principales bien definidas y desacopladas:

- **Sistema distribuido:** implementa la lógica de coordinación de los nodos de un *cluster* y el mantenimiento de sus datos.

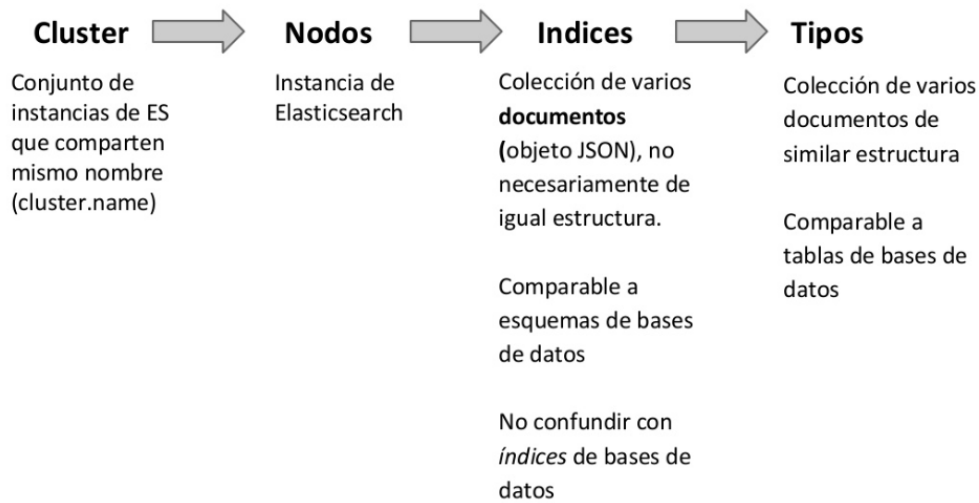


Figura 2.1: Glosario sistema distribuido

- **Motor de búsqueda:** proporciona las funcionalidades de indexación y búsqueda de documentos.

Se trata de un motor de búsqueda, orientado a documentos, basado en Apache Lucene pero expone su funcionalidad a través de una interfaz REST (Representational State Transfer, o Transferencia de Estado Representacional) recibiendo y enviando datos en formato JSON (JavaScript Object Notation) ocultando los detalles internos de Lucene. Esta interfaz permite que pueda ser utilizada por cualquier plataforma no solo Java, puede usarse desde Python, .NET, PHP o incluso desde un navegador con JavaScript. Es persistente, es decir, que lo que indexemos en la base de datos sobrevivirá a un reinicio del servidor. Las principales características de esta herramienta son:

- **Datos en tiempo real:** Elasticsearch disponibiliza los últimos cambios realizados sobre los datos en tiempo real (con muy baja latencia).
- **Distribuido y escalable horizontalmente:** lo que nos permite ir creciendo conforme lo hagan nuestras necesidades.
- **Alta Disponibilidad:** los *clusters* de Elasticsearch son capaces de detectar y eliminar nodos que estén fallando y reorganizarse a sí mismos para asegurar que los datos estén a salvo y permanezcan accesibles.

- **Multi-tenancy:** Un *cluster* de Elasticsearch puede alojar múltiples índices que pueden ser consultados de manera independiente. También permite definir índices online.
- **Búsquedas full-text:** Elasticsearch se basa en Lucene para sus capacidades de búsqueda de texto, soportando geolocalización, autocompletado ...
- **Orientado a documentos:** Las entidades se almacenan en Elasticsearch como documentos JSON estructurados. Todos los campos son indexados por defecto y todos los índices pueden ser usados en una misma consulta.
- **Gestión de conflictos:** provee mecanismos (optimistic version control, o control optimista de versiones) para asegurar que los datos no se pierdan debido a cambios simultáneos sobre un mismo documento realizados por diferentes procesos.
- **Sin esquemas:** permite trabajar sin esquemas que definan la estructura de los datos.
- **API Restful:** Elasticsearch proporciona un API Restful sobre JSON. Además ofrece APIs para otros lenguajes como Java.

2.2. Kibana



Kibana es una herramienta *open-source* perteneciente a Elastic, que permite visualizar y explorar datos que se encuentran indexados en Elasticsearch. Este *framework* también es conocido por el *stack* ELK [13]: elasticsearch, logstash & kibana. Dichas herramientas facilitan la ingesta de datos de fuentes distribuidas, indexan la información y posteriormente la representan gráficamente. Kibana es una herramienta *front end* diseñada para funcionar de forma conjunta con Elasticsearch con el objetivo de que los datos sean más fáciles de entender e interpretar a través de representaciones gráficas. El funcionamiento básico hace uso de *queries* que a su vez emplean agregaciones de datos y filtros. Existen dos tipos de agregaciones:

Request:

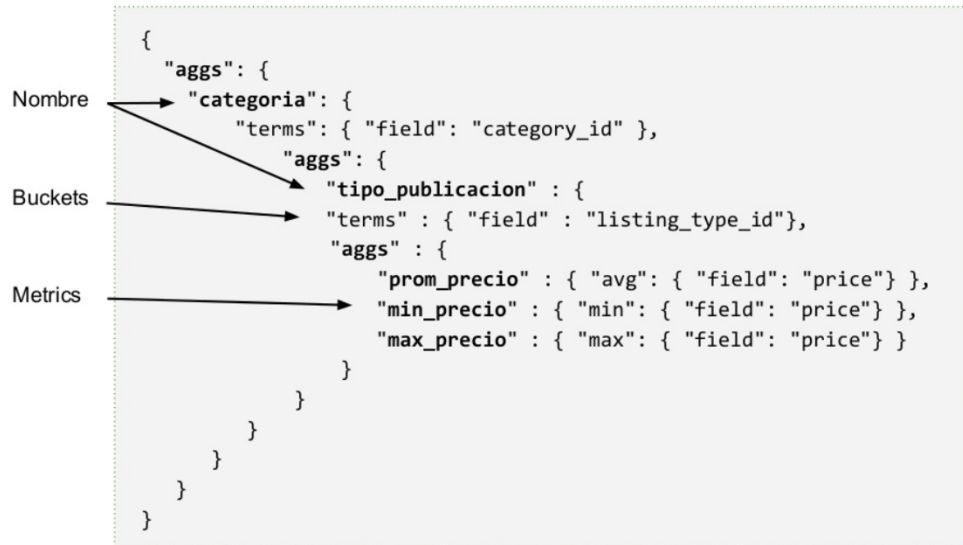


Figura 2.2: Ejemplo de agregaciones

- **Buckets:** En una búsqueda concreta es posible separar los documentos que cumplen una determinada condición. Es equivalente a “group by” de SQL.
- **Metrics:** son funciones matemáticas aplicadas sobre los buckets y algunas funciones equivalentes en SQL son: *COUNT*, *AVG*, *SUM*, *MAX*, *etc.*

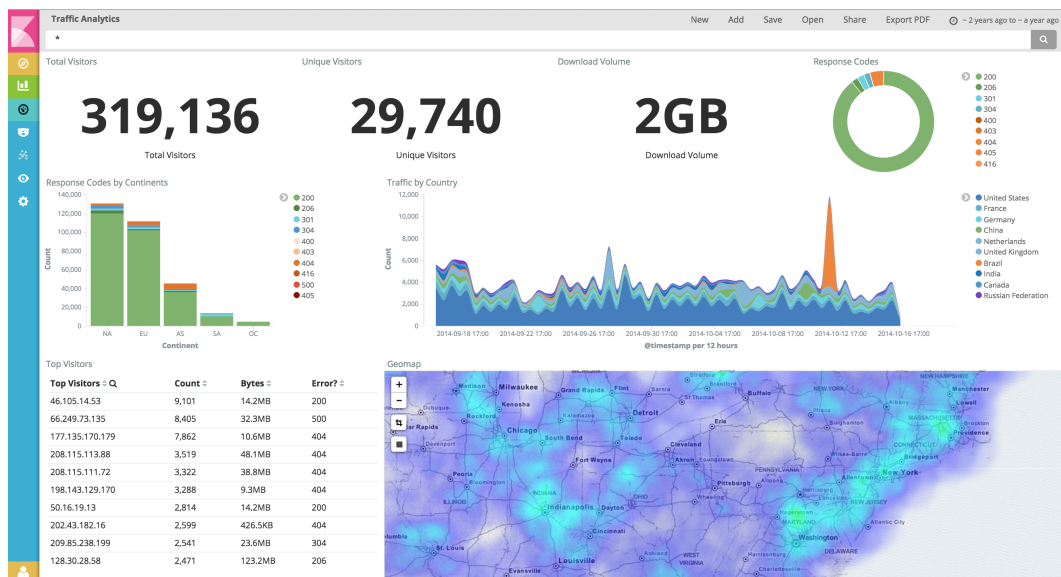


Figura 2.3: Diseño Kibana 5

Un caso común es utilizar una combinación entre buckets y metrics. Sin embargo se puede añadir más complejidad y definir diferentes niveles de buckets lo que permite agrupar los datos por más de un criterio y obtener resultados concretos.

Kibana ofrece diferentes tipos de representaciones gráficas como tartas, líneas, barras, tablas, etc. Además los desarrolladores pueden implementar sus propios *plugins* y crear nuevos tipos de representaciones. Se dispone de la posibilidad de customizar los gráficos, crear cuadros de mando, guardarlos en Elasticsearch y compartirlos.

2.3. JavaScript

JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript, y fue desarrollado por Brendan Eich, trabajador de Netscape en 1995. Este lenguaje es orientado a objetos, imperativo y con tipado débil y dinámico.



La primera versión de JavaScript fue un completo éxito y Netscape Navigator 3.0 ya incorporaba la siguiente versión del lenguaje, la versión 1.1. Al mismo tiempo, Microsoft lanzó JScript con su navegador Internet Explorer 3. JScript era una copia de JavaScript al que le cambiaron el nombre para evitar problemas legales. Para evitar una guerra de tecnologías, Netscape decidió que lo mejor sería estandarizar el lenguaje JavaScript. De esta forma, en 1997 se envió la especificación JavaScript 1.1 al organismo *ECMA* (European Computer Manufacturers Association, o Asociación Europea de Fabricantes de Ordenadores). Por este motivo, algunos programadores prefieren la denominación ECMAScript para referirse al lenguaje JavaScript. De hecho, JavaScript no es más que la implementación que realizó la empresa Netscape del estándar ECMAScript.

Javascript es un lenguaje de programación que surgió con el objetivo inicial de programar ciertos comportamientos sobre las páginas web, respondiendo a la interacción del usuario y la realización de automatismos sencillos. En ese contexto podríamos decir que nació como

un "lenguaje de scripting" del lado del cliente, sin embargo, hoy JavaScript es mucho más. Las necesidades de las aplicaciones web modernas y el HTML5 han provocado que el uso de JavaScript que encontramos hoy haya llegado a unos niveles de complejidad y prestaciones tan grandes como otros lenguajes de primer nivel. Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del *DOM* (Document Object Model, o modelo de objetos del documento).

En junio de 2015 se cerró y publicó el estándar ECMAScript 6 (última versión hasta la fecha) con un soporte irregular entre navegadores y que dota a JavaScript de características avanzadas que se echaban de menos y que son de uso habitual en otros lenguajes como, por ejemplo, módulos para organización del código, verdaderas clases para programación orientada a objetos, expresiones de flecha, iteradores, generadores o promesas para programación asíncrona.

2.4. AngularJS



AngularJS es un *framework* web abierto de JavaScript, mantenido por Google y que permite la creación de SPA (Single Page Applications, o Aplicaciones de Página Única). Es un proyecto de código abierto, que contiene un conjunto de librerías útiles para el desarrollo de aplicaciones web y propone una serie de patrones de diseño para llevarlas a cabo. En pocas palabras, es lo que se conoce como un *framework* para el desarrollo sobre el lenguaje JavaScript con programación del lado del cliente. AngularJS permite crear HTML enriquecido gracias a las directivas. Una de las características más impactantes es el *two-way data binding*, o enlazado de datos de forma bidireccional. Esta funcionalidad, enlaza un modelo de datos con su representación en la vista. Por lo tanto, cualquier cambio realizado en la vista se verá reflejado en el modelo y viceversa. Las principales características de este framework son:

- **Scopes:** Los *scopes* (ámbitos de una variable) son los distintos contextos de ejecución sobre los que trabajan las expresiones de AngularJS. En dichos ámbitos se guarda la información de los modelos que se representan en la vista y también atributos que se utilizan para manejar la lógica de la misma. Estos ámbitos se manejan principalmente desde los controllers y directivas.
- **Controllers:** Los controllers son los encargados de inicializar y modificar la información que contienen los ámbitos de las variables y funciones dependiendo de las necesidades de la aplicación. También es posible declarar funciones en el *scope* que se podrán utilizar más tarde o ser llamadas desde la vista.
- **Client-side template:** El sistema de plantillas en AngularJS es diferente del utilizado en otros *frameworks*. Por lo general es el servidor el encargado de mezclar la plantilla con los datos y devolver el resultado al navegador. En AngularJS el servidor proporciona los contenidos estáticos (plantillas) y la información que se va a representar (modelo) pero es el cliente el encargado de mezclar la información del modelo con la plantilla para generar la vista.
- **Data binding:** Con AngularJS podemos sincronizar el modelo y la vista automáticamente utilizando ciertas directivas del *framework*. Esta sincronización es bidireccional, es decir, la información se sincroniza tanto si cambia el valor en la vista como si lo hace el valor en el modelo. Dicha funcionalidad recibe el nombre de *two-way data binding* y es una de las principales características de AngularJS.
- **Directivas:** Las directivas son el plato fuerte de AngularJS. Mediante el uso de las mismas podemos extender la sintaxis de HTML y darle el comportamiento deseado. Se pueden crear directivas a nivel de elemento, de atributo, de clase y de comentario. Una de las cosas más interesantes de las directivas es la posibilidad de declararlas a nivel de elemento, lo que permite a los usuarios crear nuevas etiquetas de HTML que facilitan la creación de componentes reutilizables.

2.5. D3.js



La librería D3.js recibe dicho nombre por las siglas *Data-Driven Documents* y permite producir, a partir de los datos de entrada, gráficos interactivos. Se trata de una herramienta importante, conforme a las normas W3C que utiliza las tecnologías habituales SVG, JavaScript y CSS para la visualización de datos.

Dicha biblioteca admite un control muy amplio del resultado final lo que da lugar a la creación de algunos diagramas bastante satisfactorios una vez superada la curva de dificultad que presenta al principio. Es de código abierto, lo que da lugar a las abundantes bibliotecas basadas en D3.js, mucho más simples y fáciles de usar pero más restrictivas.

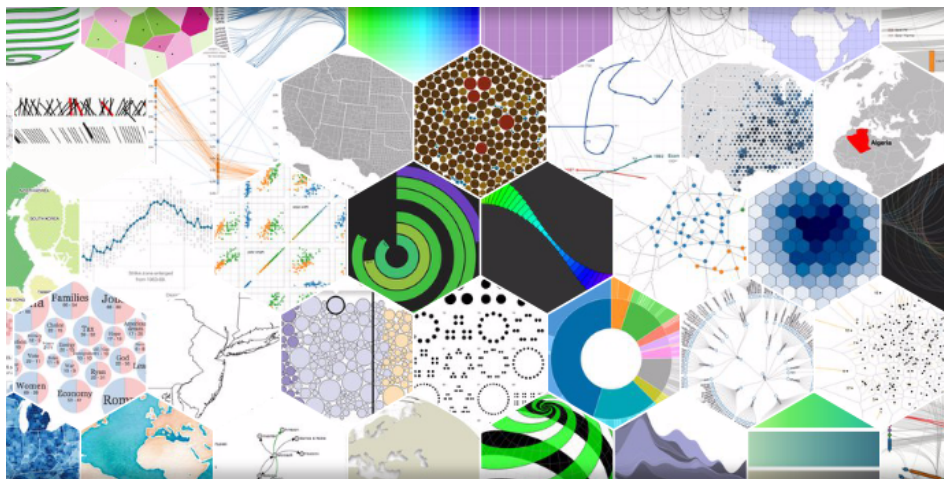


Figura 2.4: Gráficos D3

2.6. C3.js

C3.js [7] es una biblioteca gráfica muy flexible de JavaScript basada en D3.js (más generica y complicada) que permite cambiar la configuración (tipo de gráfica, zoom, cargar nuevos datos,

etc.) una vez que la representación haya sido renderizada. C3.js hace que la representación de datos sea muy fácil ya que envuelve el código que utiliza D3.js para construir los gráficos evitando que el usuario tenga que escribir sentencias en JavaScript. Existen multitud de opciones que nos permiten personalizar las representaciones (colores, etiquetar los ejes, zoom, añadir otro eje de ordenadas, crear regiones, utilizar cuadrícula, transformaciones, animaciones, etc.).

2.7. JSON

JSON, acrónimo de JavaScript Object Notation, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML. Su estructura está basada en cómo se crea un objeto JavaScript, formado por tuplas *clave-valor* para representar cada propiedad de una entidad o corchetes para los arrays.

Una de las supuestas ventajas de JSON sobre XML como formato de intercambio de datos es que es mucho más sencillo escribir un analizador sintáctico (parser) de JSON. En JavaScript, un texto JSON se puede analizar fácilmente usando la función `eval()`, lo cual ha sido fundamental para que JSON haya sido aceptado por parte de la comunidad de desarrolladores AJAX, debido a la ubicuidad de JavaScript en casi cualquier navegador web. JSON se emplea habitualmente en entornos donde el tamaño del flujo de datos entre cliente y servidor es de vital importancia, cuando la fuente de datos es explícitamente de fiar y donde no es importante el no disponer de procesamiento XSLT (transformaciones de lenguaje extensible de hojas de estilo) para manipular los datos en el cliente.

El formato JSON es el formato que se utiliza por defecto en la comunicación y transferencia de datos entre Elasticsearch y Kibana debido a sus propiedades. Sin embargo Elasticsearch también acepta otros formatos como cURL.

2.8. NodeJS y NPM

NodeJS es una tecnología que se apoya en el motor de JavaScript V8 (creado por Google) para permitir la ejecución de programas hechos en JavaScript en un ámbito independiente del navegador. A veces se hace referencia a NodeJS como JavaScript del lado del servidor, pero es

mucho más.

La característica más importante de NodeJS, y que ahora otra serie de lenguajes están aplicando, es la de no ser bloqueante. Es decir, si durante la ejecución de un programa hay partes que necesitan un tiempo para producirse la respuesta, NodeJS no para la ejecución del programa esperando que esa parte acabe, sino que continúa procesando las siguientes instrucciones. Cuando el proceso lento termina, entonces efectúa las sentencias pendientes de realizar con los resultados recibidos.

Dicha característica hace que el lenguaje sea muy rápido y que permita la creación de programas de red altamente escalables, como por ejemplo, servidores web. Existen muchas aplicaciones de NodeJS, aunque la más famosa es la de atender comunicaciones por medio de *sockets*, necesarias para las aplicaciones en tiempo real.



NPM (Node Package Manager) es el manejador de paquetes por defecto para NodeJS. Con este manejador, los desarrolladores pueden crear, compartir y reutilizar módulos en sus aplicaciones. Permite utilizar desde pequeños módulos hasta aplicaciones completas de un catálogo que a día de hoy es bastante extenso.

NPM utiliza el archivo *package.json* para almacenar todos los datos relevantes de un programa, es decir, los metadatos. En dicho archivo debemos declarar todos los módulos que son necesarios para el correcto funcionamiento de nuestra aplicación, o dicho de otra forma, las dependencias del programa. De esta forma cuando otra persona utilice nuestro código en primer lugar tendría que lanzar el comando `npm install` para descargar en local los archivos que requiera la aplicación. En este proyecto únicamente es necesario declarar la dependencia de C3.js y la versión deseada ya que internamente tiene otra dependencia con D3.js.

```
"dependencies" : {  
  "c3": "0.4.11"  
}
```

2.9. Elasticdump

Elasticdump es un módulo de NPM que sirve para enviar una entrada a una salida. Típicamente se utiliza para para cargar datos a un *cluster* de Elasticsearch o, por el contrario, descargar los datos de un *cluster* a un archivo local. El funcionamiento es muy sencillo y se realiza a través del interprete de comandos. Un posible ejemplo de uso sería el siguiente:

```
elasticdump \  
--input=path_to_file/my_data.json \  
--output=http://localhost:9200/my_index \  
--type=data
```

2.10. HTML

HTML5 es la quinta revisión importante del lenguaje básico de la *World Wide Web*. HTML5 especifica dos variantes de sintaxis para HTML: HTML (text/html), la variante conocida como HTML5 y una variante XHTML conocida como sintaxis XHTML5 que deberá ser servida como XML. Esta es la primera vez que HTML y XHTML se han desarrollado en paralelo.



Es un lenguaje sencillo, basado en etiquetas (rodeadas por corchetes angulares), que definen la forma en la que se despliega o se muestra el contenido al usuario. Se estructura típicamente en dos partes, una primera parte que es la cabecera *<head>*, que suele contener información del documento que se presenta, y un cuerpo (etiqueta *<body>*), que define el contenido del documento. A su vez dentro de estas etiquetas o elementos, encontramos atributos, que suelen ser tuplas *nombre-valor*. Un claro ejemplo es el atributo “*class*” que es fundamental para la integración con Bootstrap, de tal forma que se puedan identificar cuáles son los estilos a aplicar a cada elemento de la página web. Algunas de las principales novedades de HTML5 son:

- Mejora el elemento *canvas*, capaz de renderizar elementos 3D.
- Almacenamiento local (LocalStorage) de datos en el lado del cliente pero debemos ver si el navegador en el que estamos trabajando es compatible.
- Mejoras en los formularios. Nuevos tipos de datos (eMail, number, url, datetime ...) y facilidades para validar el contenido sin JavaScript.
- Elementos importantes: **audio** y **video** sirven para incrustar un contenido multimedia de sonido o de vídeo, respectivamente. Sin duda uno de los añadidos más interesantes, ya que permite reproducir/controlar vídeos y audios sin necesidad de *plugins* como el de Flash. Se tratan de manera totalmente nativa como cualquier otro elemento, por ejemplo se pueden incluir enlaces o imágenes dentro de un vídeo.

2.11. Bootstrap



Bootstrap

Bootstrap es un conjunto de herramientas de código abierto, desarrollado y mantenido por Twitter, que permite hacer nuestra página HTML/CSS responsiva, es decir, Bootstrap permite que una página web se adapte al tamaño del navegador o el dispositivo en que está siendo mostrada. Esta funcionalidad la otorga el sistema de celdas de Bootstrap que permite una mejor maquetación de nuestra página web. El *framework* trae varios elementos con estilos predefinidos fáciles de configurar: botones, menús desplegables, formularios incluyendo todos sus elementos e integración jQuery para ofrecer ventanas y *tooltips* dinámicos. Se ha utilizado esta tecnología para construir el sitio web del proyecto.

2.12. CSS



CSS o Cascading Style Sheets es un lenguaje que define la presentación de un documento escrito en HTML. Al igual que la codificación HTML, el W3C es el encargado de elaborar el estándar que finalmente han optado por seguir todos los navegadores. La sintaxis de la que hace uso CSS es muy sencilla, se basa en asignar un valor a una propiedad. La idea fundamental de CSS reside en separar la estructura del documento de la vista del mismo. La información de presentación se puede encontrar en el mismo fichero HTML o en un documento separado.

Algunas de las principales novedades de CSS3 son:

- **Bordes:** border-color, border-image, border-radius, box-shadow.
- **Fondos:** background-origin, background-clip, background-size or multiple backgrounds.
- **Colores:** HSL, HSLA, RGBA, Opacidad.
- **Texto:** text-shadow, text-overflow.

2.13. LESS

LESS es un lenguaje dinámico de hojas de estilo diseñado por Alexis Sellier. Su primera versión fue escrita en Ruby, sin embargo, en las versiones posteriores, se abandonó el uso de Ruby y se lo sustituyó por JavaScript. La principal diferencia entre LESS y otros precompiladores CSS es que LESS permite la compilación en tiempo real a través de LESS.js por el navegador. LESS se puede ejecutar en el lado del cliente y en el lado del servidor, o se puede compilar en CSS sin formato. LESS permite la definición de variables a través del carácter arroba (@). La asignación de variables se hace con dos puntos (:). Durante la traducción, los valores de las variables se insertan en el documento CSS de salida.

```
CSS ▾  
  
/* Less Code */  
@heading-color: rgb(99, 200, 200);  
h1 {  
  color: @heading-color;  
  font-weight: 300;  
}  
  
/* Compiled CSS code */  
h1 {  
  color: #63c8c8;  
  font-weight: 300;  
}
```

Figura 2.5: Variable LESS

2.14. Git



Git es un sistema de control de versiones de código libre y abierto, creado por Linus Torvalds para mantener el desarrollo del núcleo de Linux. Por tanto, está diseñado para controlar cualquier tipo de proyectos independientemente de su magnitud. Un *commit* es una foto de nuestros archivos en un determinado momento. Estos incluyen un identificador, todos los cambios respecto al *commit* anterior y una referencia al mismo. De esta manera, siempre que queramos, podremos retroceder hasta una versión anterior de nuestro código. También nos permite tener varias versiones en paralelo o ramas de nuestros proyectos. Estas características resultan muy útiles para trabajos en equipo.

2.15. GitHub



GitHub es una plataforma de desarrollo colaborativo de software para alojar proyectos utilizando el sistema de control de versiones Git. Utiliza el *framework* Ruby on Rails por GitHub, Inc. (anteriormente conocida como Logical Awesome). Desde enero de 2010, GitHub opera bajo el nombre de GitHub, Inc. El código de los usuarios se almacena de forma pública, aunque también se puede hacer de forma privada, creando una cuenta de pago.

Este servicio de alojamiento de repositorios es utilizado por muchas de las grandes empresas para alojar sus desarrollos públicos, tales como *SDK* (software development kit, o kit de desarrollo de software), librerías, ejemplos, etc. Github ha lanzado un servicio denominado GitHub Pages que permite crear páginas webs para nuestros proyectos, usuario y organizaciones de forma gratuita directamente desde un repositorio a través del dominio *github.io* que opera con el protocolo *HTTPS* (Hypertext Transfer Protocol Secure, o protocolo seguro de transferencia de hipertexto).

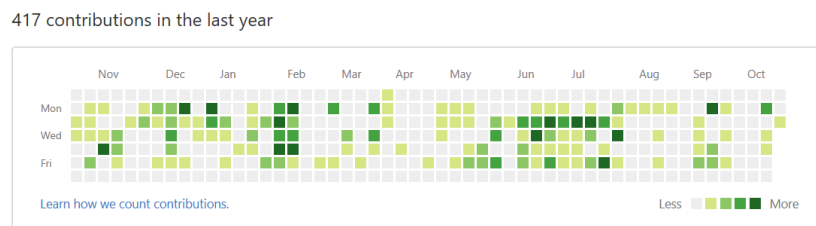


Figura 2.6: Ejemplo contribuciones GitHub

2.16. Metodología SCRUM

Este modelo fue identificado y definido por Ikujiro Nonaka e Hirotaka Takeuchi a principios de los 80, al analizar cómo desarrollaban los nuevos productos las principales empresas de manufactura tecnológica. En su estudio, Nonaka y Takeuchi compararon la nueva forma de

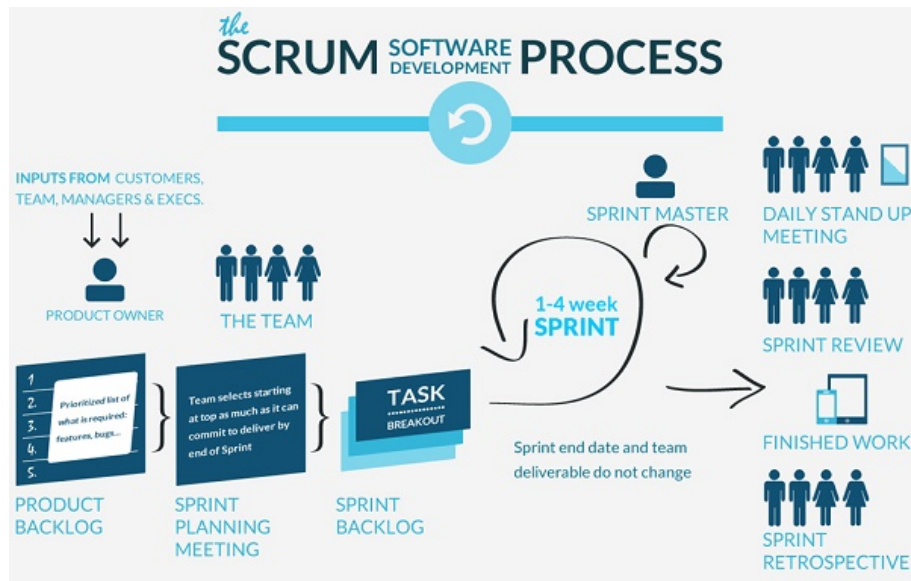


Figura 2.7: Esquema del modelo Scrum

trabajo en equipo, con el avance en formación de melé (scrum en inglés) de los jugadores de rugby, a raíz de lo cual quedó acuñado el término “scrum” para referirse a ella.

Scrum es un proceso en el que se aplican de manera regular un conjunto de buenas prácticas para trabajar colaborativamente, en equipo, y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos.

En Scrum se realizan entregas parciales y regulares del producto final, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad, la flexibilidad y la productividad son fundamentales.

Scrum también se utiliza para resolver situaciones en que no se está entregando al cliente lo que necesita, cuando las entregas se alargan demasiado, los costes se disparan o la calidad no es aceptable, cuando se necesita capacidad de reacción ante la competencia, cuando la moral de los equipos es baja y la rotación alta, cuando es necesario identificar y solucionar ineficiencias sistemáticamente o cuando se quiere trabajar utilizando un proceso especializado en el desarrollo de producto.

2.17. Herramientas para representar datos

Actualmente podemos encontrar un conjunto de herramientas orientadas al análisis y visualización de datos incluso en tiempo real. Algunos de esos proyectos se explican a continuación.

Bitergia

Bitergia investiga cómo innovar en el análisis cuantitativo en profundidad de proyectos de software libre, en los que todo el proceso de desarrollo se realiza en herramientas que transmiten información accesible a cualquier persona interesada. Bitergia, que toma como base de su negocio los datos ofrecidos por las herramientas de desarrollo, ha producido una serie de herramientas que permiten llevar a cabo la extracción y posterior análisis de dicha información. El objetivo es que los clientes puedan tomar decisiones basadas en datos objetivos y no en percepciones subjetivas del proyecto.

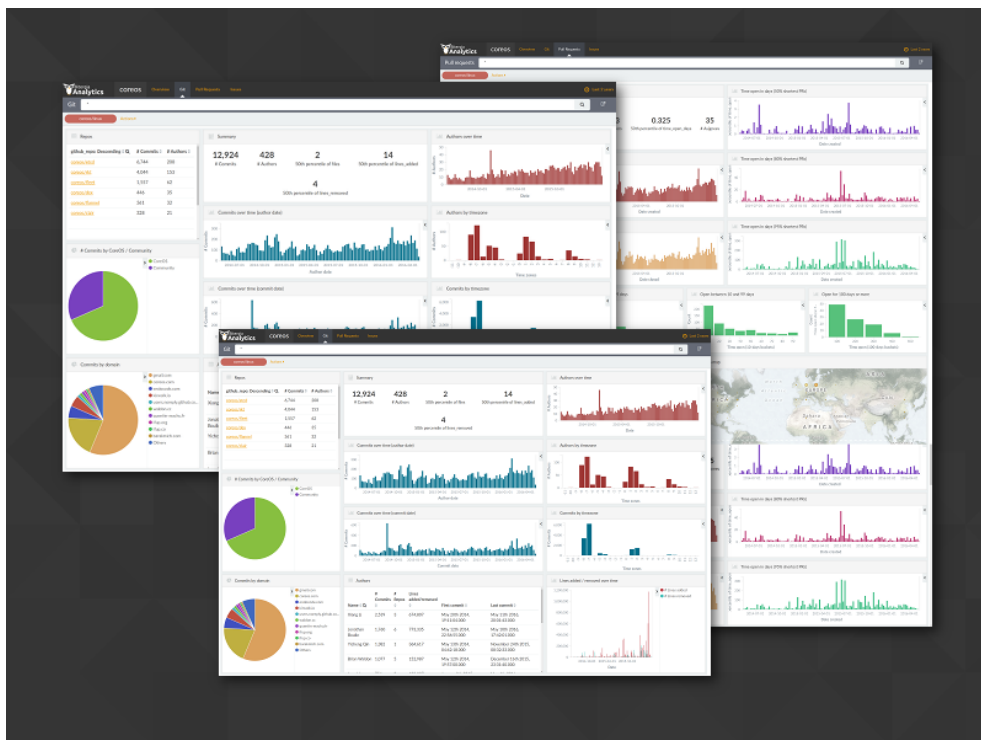


Figura 2.8: Bitergia

FreeBoard

FreeBoard es un producto de Bug Labs, una Startup de Nueva York. Es una plataforma Open Source que trata de convertir el *IoT* (Internet of Things, o internet de las cosas) en un entorno mucho más sencillo, simple y accesible para todo el mundo. Esta plataforma ofrece a los usuarios una forma sencilla, mediante un único clic, de publicar los datos de un dispositivo con conexión a Internet a su propia página web creando sencillos cuadros de mando.

Cyfe

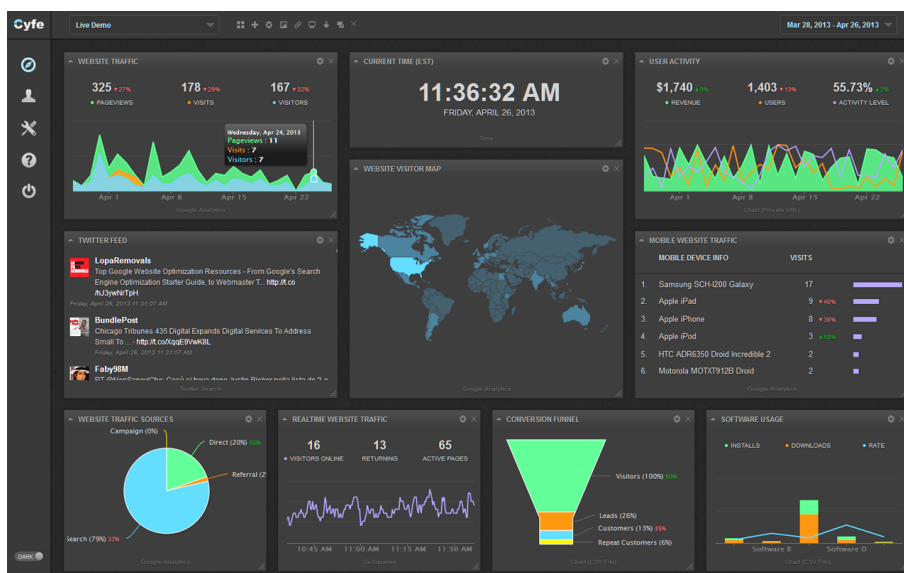


Figura 2.9: Cuadro de mando Cyfe

La mayoría de bloggers o administradores de medios sociales se encuentran con el problema de obtener los análisis en un solo lugar que provengan de los distintos servicios web. Cyfe es un servicio con el que podemos construir un cuadro de mando a medida usando como fuentes de información un buen número de servicios disponibles en la red como Google Analytics, Salesforce, AdSense, MailChimp, Amazon, Facebook, WordPress, Zendesk, Twitter o Pingdom.

La idea es concentrar toda la información en un único lugar y, de un vistazo, obtener información gráfica de la disponibilidad de nuestra página web, los seguidores que tenemos en Twitter o el número de usuarios que acceden al *feed* de nuestro blog. Las principales caracte-

terísticas de la herramienta son la unificación de las métricas, la personalización y la obtención de estadísticas en tiempo real.

Grafana

Grafana es una herramienta *open-source* que trabaja con bases de datos como *Graphite*, *InfluxDB*, *OpenTSDB* e incluso *Elasticsearch*. Permite a los usuarios crear y editar cuadros de mando de una forma rápida y sencilla. La principal diferencia con Kibana es que Grafana se centra en representar datos basados en series temporales de métricas específicas tales como la utilización de *CPU* (Central Processing Unit, o unidad central del procesamiento) o E/S. Sin embargo, en contrapartida, Grafana no permite la exploración o búsqueda de datos.



Figura 2.10: Visualizaciones en Grafana

Al igual que ocurre con Kibana, al tratarse de un software de código abierto, los desarrolladores pueden desplegar *plugins* y publicarlos para que los usuarios o clientes de la herramienta puedan utilizarlos. Entre las visualizaciones oficiales podemos encontrar diagramas, histogramas, tablas, textos y un largo etcétera.

Capítulo 3

Diseño e implementación

Este es el capítulo central de la memoria ya que en esta sección se desarrollará el proceso seguido para lograr el resultado final del proyecto. Se analizará el uso y el desarrollo de la aplicación desde un punto de vista incremental, guiando al lector por cada una de las etapas que se han atravesado.

3.1. Modelo de desarrollo

Para este proyecto se ha seguido la metodología SCRUM mencionada en el segundo capítulo, consiguiendo un desarrollo incremental con reuniones periódicas y pasando por varias iteraciones. A continuación se describen los ciclos que se han realizado con sus respectivos requisitos, implementaciones y resultados obtenidos.

- **Iteración 0:** Se trata de una fase previa antes de comenzar a desarrollar el proyecto en sí. El objetivo de esta etapa es conocer y aprender las tecnologías que se vayan a utilizar (JavaScript y AngularJS), jugar con ejemplos sencillos y entender conceptos básicos.
- **Iteración 1:** Esta etapa tiene como objetivo conocer el funcionamiento de Elasticsearch, aprender a utilizar la herramienta e integrar la base de datos con una aplicación de ejemplo de AngularJS. Paralelamente se busca información sobre bibliotecas gráficas para que posteriormente se elija la más adecuada para este proyecto.

- **Iteración 2:** En una segunda fase se da un paso más allá para descubrir Kibana, instalando la última versión disponible para aprender a trabajar con la herramienta, entender su funcionamiento y generar diversas visualizaciones. Por otro lado se considera la opción de programar una herramienta desde cero que acabe siendo un *dashboard* y ofrezca funcionalidades parecidas a Kibana.
- **Iteración 3:** En la tercera fase tras haber pasado por una fase de documentación con Kibana y haber leído sobre otras alternativas como Caravel o Jupyter. Se decide que finalmente se va a utilizar Kibana para la creación de un *plugin* con representaciones gráficas alternativas a las ofrecidas por defecto por el *framework* utilizando C3.js. Se construye el primer *widget* genérico.
- **Iteración 4:** El objetivo de la cuarta iteración es desarrollar un *widget* propio que utilice los gráficos ofrecidos por C3.js y que permita cierta personalización por parte del usuario como por ejemplo cambiar el color, el tipo de gráfica, el nombre de los datos representados, que admita series de tiempo, etc. También se pretende mejorar la estética del programa y conservar la interfaz dada por defecto (leyenda, ajuste eje vertical, grid, etc.). Por otro lado se crea una página web oficial del proyecto y se trata de dar visibilidad a la aplicación en GitHub para que otros usuarios puedan utilizarla.

3.2. Iteración 0. Estudio previo

3.2.1. Objetivos

En este primer ciclo se define el objetivo principal y el alcance del proyecto así como las principales tecnologías que se van a utilizar para llevarlo a cabo. Realmente no se trata de un *sprint* como tal, sino una fase previa que es necesaria para el posterior desarrollo del proyecto. Los principales objetivos de este ciclo son:

1. Aprender JavaScript y refrescar conocimientos sobre tecnologías web.
2. Aplicar lo aprendido creando una aplicación con AngularJS.

3.2.2. Aprendizaje de JavaScript y AngularJS

El plan de estudios de la rama de sistemas de telecomunicación no contempla la enseñanza de JavaScript. Por lo tanto el primer paso lógico para lograr el objetivo del proyecto es aprender de forma autónoma dicho lenguaje de programación usando los medios disponibles.

Una lectura obligatoria, o al menos muy recomendada, es “Eloquent JavaScript” [1] para todo aquel que quiera aprender JavaScript desde cero. Además el autor se ha tomado la molestia de publicar una copia gratuita del libro en versión digital (formato PDF), algo que se agradece bastante. Otra lectura interesante y que es realmente recomendable para los futuros programadores es “Speaking JavaScript” [2] ya que es un libro bastante práctico en el que se van alternando conceptos teóricos y ejemplos prácticos sencillos. Este libro puede encontrarse tanto en formato físico como en varios formatos digitales. En cuanto al *framework* AngularJS, en la red se pueden encontrar montones de tutoriales, además de la propia página web oficial de AngularJS [8] por supuesto, que ayudarán a entender su funcionamiento básico.

Otra forma de mejorar nuestras habilidades en la programación es a través de los cursos online que ofrecen algunas plataformas muy conocidas como por ejemplo *codecademy* y *code school* [11] o a través de los tutoriales de *W3Schools* [12]. También se debe mencionar que en la página oficial de AngularJS se encuentra disponible un video-tutorial muy entretenido que

enseña el uso de algunas directivas y conceptos como el *two-way data binding* tan característicos de este *framework*.

Una vez que nos hayamos documentado sobre las nuevas tecnologías web a través de las lecturas recomendadas (no hace falta que sean libros completos), hayamos recordado y repasado otros conceptos vistos en asignaturas impartidas a lo largo de la carrera y hayamos practicado con los tutoriales o cursillos, nos disponemos a afrontar el reto de construir una aplicación para la visualización de datos de forma gráfica.

3.2.3. Desarrollo primera aplicación

El objetivo final que se estableció para esta primera instancia es la creación de una aplicación sencilla en la que se utiliza tanto HTML como JavaScript utilizando AngularJS. Lo más sencillo que se nos puede ocurrir es trabajar con variables JavaScript y archivos JSON aplicando diferentes filtrados y utilizando algunas de las directivas que ofrece AngularJS.

Para esta primera aplicación se ha utilizado Yeoman. Se trata de un conjunto de herramientas compuesto por Yo, Bower y Grunt, aplicaciones construidas sobre Node.js y que llevan a cabo una serie de tareas para hacernos la vida más fácil a la hora de desarrollar aplicaciones. Con unos pocos comandos, Yeoman se encargará de crear esqueletos para aplicaciones, comprimir imágenes o concatenar unos pocos CSS por poner un ejemplo. Por si fuera poco, Yeoman puede activar un sencillo servidor web en un directorio particular para que podamos probar la aplicación recién creada. Además trabaja con los *frameworks* más utilizados como por ejemplo Backbone, Ember o AngularJS y al iniciar un nuevo proyecto Yeoman crea el código básico (esqueleto) para los modelos, vistas y controladores.



Figura 3.1: Conjunto de herramientas para desarrollo web

Create new contact

Name:

Surname:

Age:

Phone:

Search by name:

- Juan 666 200 100
- Silvia 555 800 777
- Pedro 91 700 20 20

Search by surname:

- Perez 666 200 100
- Bermejo 555 800 777
- Pastor 91 700 20 20

Figura 3.2: Añadir y filtrar contactos

Se ha decidido crear una agenda electrónica que además de que permitirá practicar lo anteriormente mencionado, nos dará la oportunidad de utilizar formularios. El funcionamiento es muy sencillo, el usuario puede consultar la información de sus contactos o bien añadir nuevos. La aplicación no es persistente, es decir, al reiniciar el servidor se borran todos los datos excepto un par de contactos que sirven de referencia guardados como variables JavaScript.

Para añadir un nuevo contacto a la agenda se ha utilizado un formulario que permite introducir los datos de la persona (nombre, apellidos, edad y teléfono). Por otro lado se implementó un buscador a través de dos filtros que permiten buscar por el nombre de la persona o por el apellido.

ContactsJS Home **Contacts** Search New

Nombre: Juan
Apellido: Perez
Edad: 23
Teléfono: 666 200 100

Nombre: Silvia
Apellido: Bermejo
Edad: 18
Teléfono: 555 800 777

Nombre: Pedro
Apellido: Pastor
Edad: 29
Teléfono: 91 700 20 20


 Momchil Stoyanov. URJC

Figura 3.3: Lista de contactos

3.3. Iteración 1. Elasticsearch y bibliotecas gráficas

3.3.1. Objetivos

Después de tener el primer contacto con las herramientas necesarias para desarrollar el proyecto, se ha dado otro paso más introduciendo la base de datos Elasticsearch. En este ciclo se va a trabajar en la integración del motor de búsqueda con AngularJS. También se introducirán las bibliotecas gráficas. Los objetivos establecidos para esta iteración son:

1. Instalación y uso de Elasticsearch.
2. Integración del motor de búsqueda con AngularJS.
3. Estudio y selección de la biblioteca gráfica.

3.3.2. Elasticsearch

En este punto del desarrollo había llegado el momento de aprender a utilizar Elasticsearch. Para ello, en primer lugar es muy recomendable visitar la página oficial en la que podemos encontrar un video introductorio que muestra el potencial de esta herramienta. También se dispone de un manual de usuario [5] en el que se explican, con todo detalle, los pasos a seguir para poder instalar Elasticsearch, configuración básica, importación de datos y lo más importante, como usar dicho software.

Una vez descargada la última versión cliente disponible (v2.3) fue necesario importar un conjunto de datos de prueba (*test data set*). Concretamente fueron aproximadamente 15 mil tweets públicos recogidos entre las 12:00 y las 12:05 (UCT+1) el día 5 de Febrero de 2015. Dicho conjunto de datos se puede descargar desde la página oficial del producto y forma parte del tutorial para principiantes. Debido a que este proyecto no está centrado en el uso de Elasticsearch no vamos a entrar en detalle sobre la configuración, obtención y almacenamiento de los datos ya que es algo que debe hacer el usuario antes de empezar a utilizar la aplicación de este TFG.

Para entender el funcionamiento se comenzó a utilizar la API básica a través del intérprete de comandos (terminal) para conocer el estado del *cluster*, mostrar la lista de índices o realizar

una consulta a través de una *query* entre otros. Elasticsearch permite almacenar datos de forma manual a través del terminal aunque es un poco más incomodo y lento. A continuación, a modo de ejemplo, se muestra la forma manual de indexar datos en Elasticsearch.

```
PUT /customer/external/1?pretty
{
  "name": "John Doe"
}
```

Con la instrucción superior se está indexando un cliente dentro del índice *customer*, de tipo *external* y se le asigna un *ID* que es igual a 1. La respuesta recibida de que todo ha ido bien es la siguiente:

```
{
  "_index" : "customer",
  "_type" : "external",
  "_id" : "1",
  "_version" : 1,
  "result" : "created",
  "_shards" : {
    "total" : 2,
    "successful" : 1,
    "failed" : 0
  },
  "created" : true
}
```

3.3.3. Integración de Elasticsearch con AngularJS



Figura 3.4: Cluster state and indexes

Una vez realizadas las primeras pruebas con el intérprete de comandos, algo bastante incomodo y que además no permite extraer todo el potencial del motor de búsqueda, llegó el momento de integrar a Elasticsearch con el *framework* AngularJS. Para ello, surgió la idea de crear una aplicación en la que se realicen consultas básicas para mostrar información genérica de la base de datos como se muestra en la figura 3.4.

Por otro lado, para empezar a trabajar con grandes cantidades de información, se ha creado una tabla en la que se representan datos personales y bancarios de 1000 usuarios. Esto ha permitido entender el funcionamiento de las *queries* y la forma de tratar o filtrar los datos que llegan de la base de datos. Por otro lado se han podido probar algunas directivas que ofrece AngularJS tales como `ngShow` o `ngRepeat` y los servicios `$http` y `$service`.

Nombre	Apellido	Edad	Balance	Email	Domicilio
Rodriguez	Flores	31	\$27,658.00	rodriguezflores@tourmania.com	986 Wyckoff Avenue, Eastvale, HI
Opal	Meadows	39	\$24,776.00	opalmeadows@cedward.com	963 Neptune Avenue, Olney, OH
Jenkins	Haney	20	\$20,203.00	jenkinshaney@qim Monk.com	740 Ferry Place, Steinhatchee, GA
Adrian	Pitts	34	\$35,883.00	adrianpitts@combogene.com	963 Fay Court, Remington, SD
Kirsten	Fox	20	\$42,374.00	kirstenfox@codax.com	330 Dumont Avenue, Walton, AK
Vega	Flynn	20	\$42,112.00	vegaflynn@accupharm.com	647 Hyman Court, Masthope, OH
Lamb	Townsend	26	\$19,087.00	lambtownsend@geeknet.com	169 Lyme Avenue, Epworth, AL
Darla	Bridges	27	\$42,039.00	darlabridges@xeronk.com	315 Central Avenue, Woodlake, RI
Harding	Hobbs	26	\$21,137.00	hardinghobbs@xth.com	474 Ridgewood Place, Heil, ND

Figura 3.5: Tabla de datos extraídos de Elasticsearch

Una cuestión importante y que ha de tenerse en cuenta cuando se desea utilizar Elasticsearch desde un navegador web es el problema CORS (Cross-Origin Resource Sharing, o Intercambio de Recursos de Origen Cruzado). Por motivos obvios de seguridad, por defecto Elasticsearch no permite realizar peticiones entre dominios. Si se intenta hacer una petición de un dominio a otro diferente (incluso dentro de la misma máquina), por norma general se obtiene un error HTTP *Forbidden 403*. La *same-origin policy* restringe la forma en que un documento o script de un sitio web interactúa con otro recurso con distinto origen. Para evitar este problema es

necesario configurar Elasticsearch para que acepte peticiones del navegador utilizando CORS. Simplemente es necesario añadir las siguientes líneas al final del archivo *elasticsearch.yml*:

```
http.cors:
enabled: true
allow-origin: /https?:\/\/localhost(:[0-9]+)?/
```

Para definir el origen de las peticiones que son aceptadas se utilizan expresiones regulares, para este ejemplo tan solo se necesita añadir el *localhost* ya que estamos trabajando en local.

3.3.4. Bibliotecas gráficas

En la tercera fase de esta iteración nos hemos centrado en elegir la biblioteca gráfica adecuada para el proyecto. Las principales alternativas que se han barajado fueron D3.js, C3.js, NVD3, Epoch y xCharts. La biblioteca D3.js quedó descartada casi desde el primer momento ya que es bastante compleja de utilizar y es necesario escribir gran cantidad de código para poder obtener las visualizaciones oportunas. Sin embargo era más razonable elegir otra biblioteca que sea algo más restrictiva pero que ofrezca funcionalidades ya implementadas.

Se ha tratado de hacer ejemplos con el resto de bibliotecas gráficas para entender su funcionamiento y las principales características que presentan. Finalmente la que más nos llama la atención, por su usabilidad y multitud de opciones de personalización de los gráficos, es C3.js.

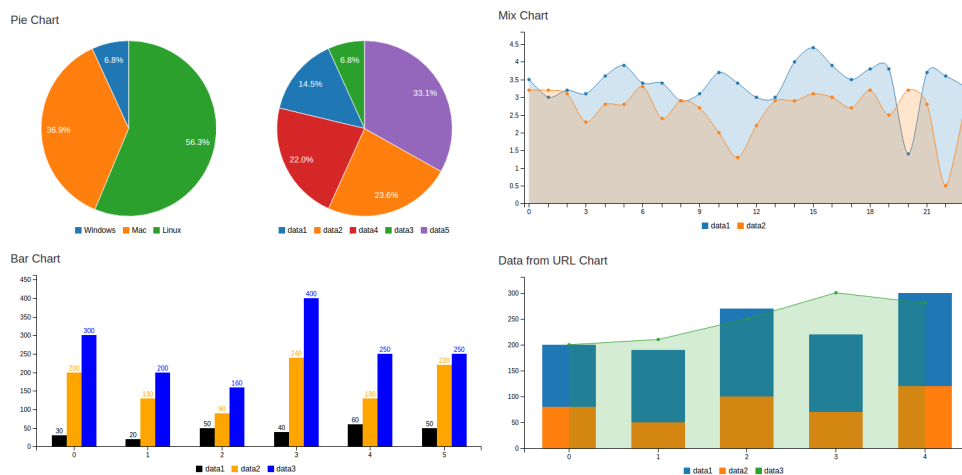


Figura 3.6: Visualizaciones con datos de Elasticsearch

3.4. Iteración 2. Dashboard propio y Kibana

3.4.1. Objetivos

Una vez elegida la biblioteca gráfica en el ciclo anterior llega la hora de crear el primer *dashboard*. En este punto del proyecto tratábamos de avanzar por dos caminos paralelos: en primer lugar desarrollar una aplicación propia de visualización de datos desde cero aprovechando la soltura obtenida en la iteración anterior al integrar la biblioteca gráfica con AngularJS y habiendo conocido el funcionamiento de Elasticsearch. Por otro lado quedaba pendiente dar una oportunidad a Kibana, una herramienta creada por los desarrolladores del motor de búsqueda Elasticsearch que sirve para representar de forma gráfica los datos almacenados en la base de datos. Para este *sprint* marcamos los siguientes objetivos:

1. Crear un *dashboard* propio.
2. Aprender a utilizar Kibana.

3.4.2. Desarrollar mi propio dashboard

En la iteración anterior se había conseguido integrar diferentes visualizaciones y representar datos estáticos en el navegador con C3.js. Para seguir avanzando en esta dirección era necesario conectar el *front end* con el *back end*, es decir, AngularJS + C3.js con Elasticsearch y además dar cierta libertad y poder al usuario para que pueda elegir los datos que quiere representar y el tipo de visualización con la correspondiente customización.

En una primera aproximación, con el objetivo de integrar Elasticsearch, las peticiones a la base de datos se realizaban de forma interna insertando *queries* dentro del código y dibujando diferentes visualizaciones con los datos que devolvía Elasticsearch. Sin embargo esta herramienta tiene poca utilidad ya que el usuario de la aplicación no debería tener que cambiar el código cada vez que quiera representar sus datos.

Para hacer que la aplicación fuera más dinámica surge la idea de crear una mezcla entre *dashboard* y generador de informes en el que se mezclan representaciones gráficas y texto plano. La idea básica es muy sencilla, el usuario parte de una página en blanco en la que puede

elegir si desea incluir una gráfica o bien texto plano. En función de su elección se despliegan diferentes menús. Para introducir texto se tiene un cuadro de texto y un botón para insertarlo. Para el caso en el que se quiera insertar una visualización, el usuario debe introducir el JSON de la *query* a Elasticsearch y posteriormente puede elegir algunas personalizaciones como el tipo de gráfica (barras, tarta o líneas), el color y si desea reutilizar el último *canvas* o crear uno nuevo.

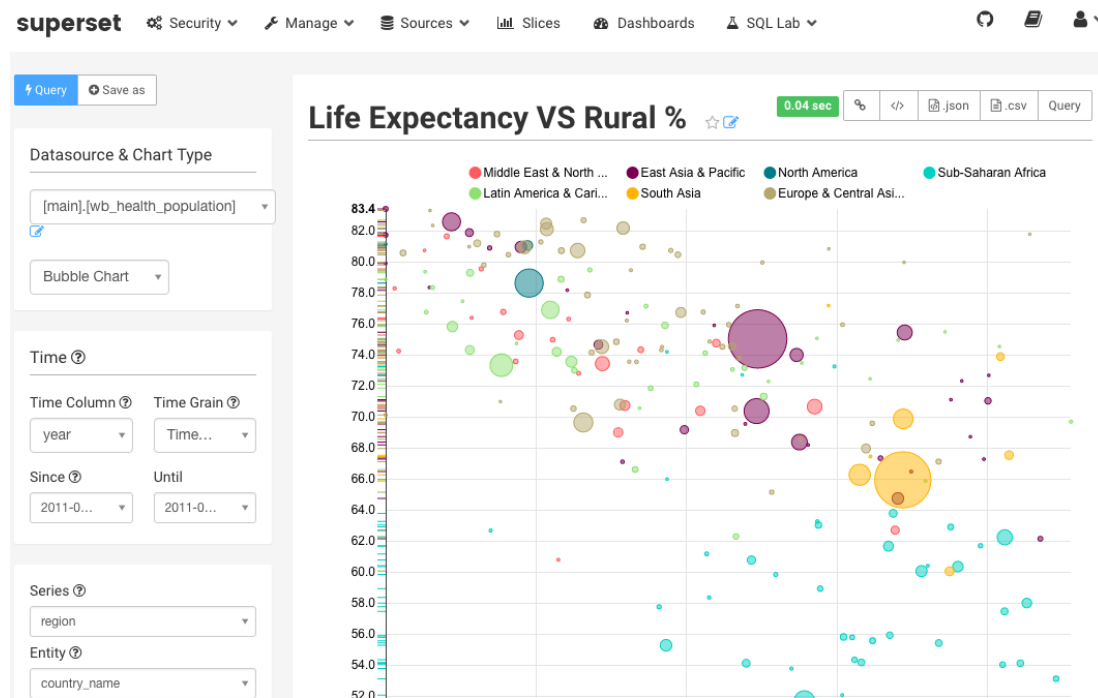


Figura 3.7: Representación gráfica con Superset

Sin embargo, una vez probado el funcionamiento de Kibana y exploradas otras opciones como pueden ser Superset [9] o Jupyter [10] me di cuenta que crear algo que tenga utilidad real va a suponer mucho esfuerzo y tiempo. La mejor opción es desarrollar un *plugin* para Kibana, ya que funciona a la perfección con Elasticsearch y ofrece diversas funcionalidades por defecto.

3.4.3. Kibana

Cuando uno busca información en la red acerca de Elasticsearch se percata enseguida de que en el 90% de los casos se relaciona la base de datos con su herramienta hermana: Kibana. De hecho Kibana tiene mayor protagonismo debido a que es lo que ve el usuario final y con lo que puede interactuar mientras que el motor de búsqueda se queda algo oculto y desapercibido

en el lado del servidor.

Instalar la última versión cliente (Kibana 4) no resultó complicado, ya que igual que ocurre con Elasticsearch los creadores del proyecto se han encargado de crear un manual [5] que explica todos los pasos necesarios. Una vez instalado el software y aprovechando que los datos indexados en Elasticsearch anteriormente se comenzaron a explorar las diferentes pestañas (descubrir, visualizar, *dashboard* y configuración) y las opciones disponibles. Una lectura muy recomendable, para todo aquel que quiera aprender a utilizar esta aplicación, es el tutorial de Tim Roes. En dicho tutorial se explican los conceptos básicos y los pasos necesarios para poder desarrollar un *plugin* propio. Una vez entendido el funcionamiento de Kibana es muy sencillo crear visualizaciones e integrarlas en un *dashboard*.

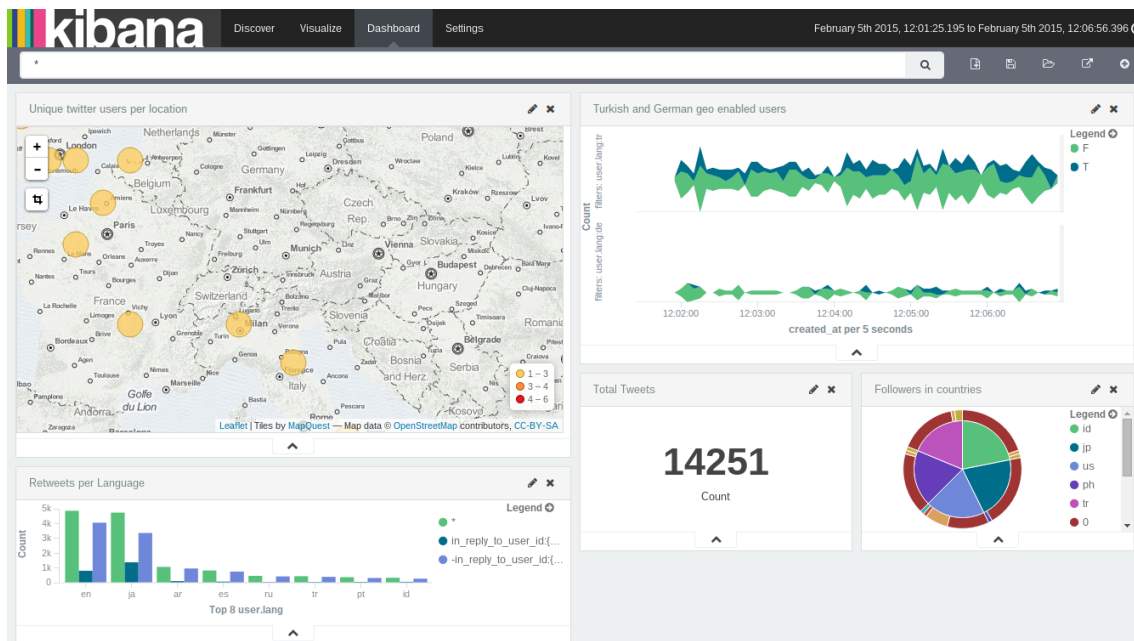


Figura 3.8: Dashboard en Kibana 4

3.5. Iteración 3. Construcción de widgets en Kibana

3.5.1. Objetivos

En esta iteración se decidió desarrollar el proyecto con ayuda de Kibana ya que es el *framework* perfecto para funcionar con Elasticsearch y además permite al desarrollador abstraerse de

algunas cuestiones como por ejemplo el intercambio de información con la base de datos o la forma de salvar las visualizaciones. Es el momento de crear un entorno de desarrollo, empezar a estudiar el código existente y tratar de programar *plugins* propios. Los objetivos de este *sprint* son los siguientes:

1. Crear un *widget* de prueba que no utilice datos.
2. Integrar la biblioteca gráfica C3.js con Kibana.

3.5.2. El primer widget

Para llevar a cabo este objetivo era necesario instalar en local la versión de desarrollador de Kibana que se encuentra alojada en GitHub [6]. Posteriormente se revisó el código que está organizado en diferentes carpetas ya que es un proyecto bastante extenso. Uno de los grandes problemas a la hora de intentar programar una aplicación que funcione con Kibana es la falta de documentación en la red debido a que es una herramienta bastante reciente. La mayoría de tutoriales y artículos explican cómo poner en funcionamiento la versión cliente a nivel de usuario. Recabar información sobre como funciona la herramienta a nivel interno es de las tareas que más tiempo han llevado de todo el proyecto.

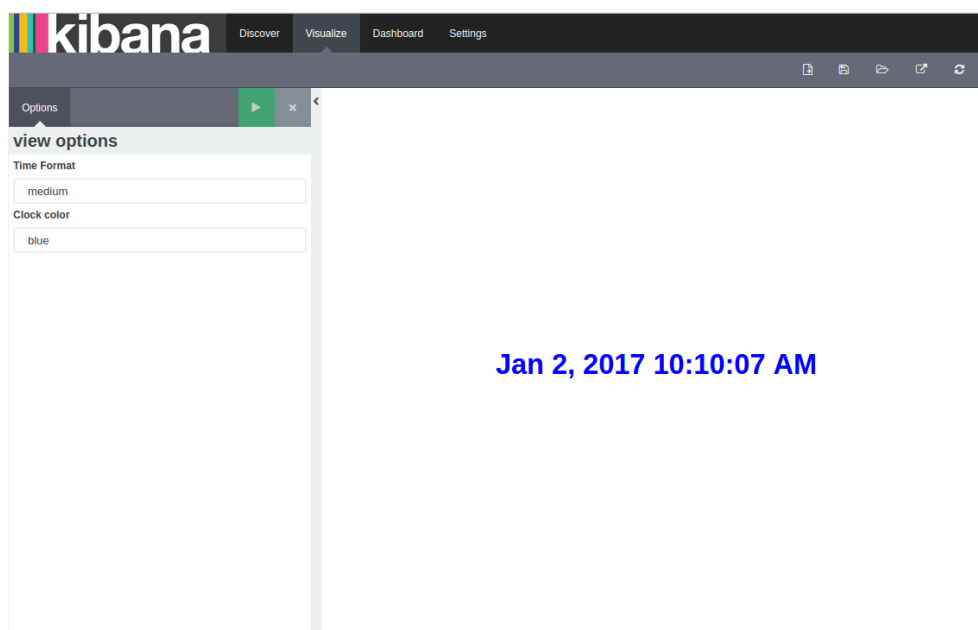


Figura 3.9: El reloj digital en Kibana 4

Finalmente fue necesario recurrir a los *plugins* que vienen por defecto con el *framework* como por ejemplo la extensión de *markdown* o la de *metrics* que son más sencillas. El primer *widget* que fue programado y comprobado que funciona fue un reloj digital que no necesita datos. Dicho reloj presenta dos personalizaciones por parte del usuario: en primer lugar poder elegir el formato de la hora según los filtros que admite AngularJS para `date object`¹. La otra customización posible es la modificación del color del texto bien introduciendo el código hexadecimal o bien el nombre del color en inglés.

3.5.3. Integración de C3.js en Kibana 5

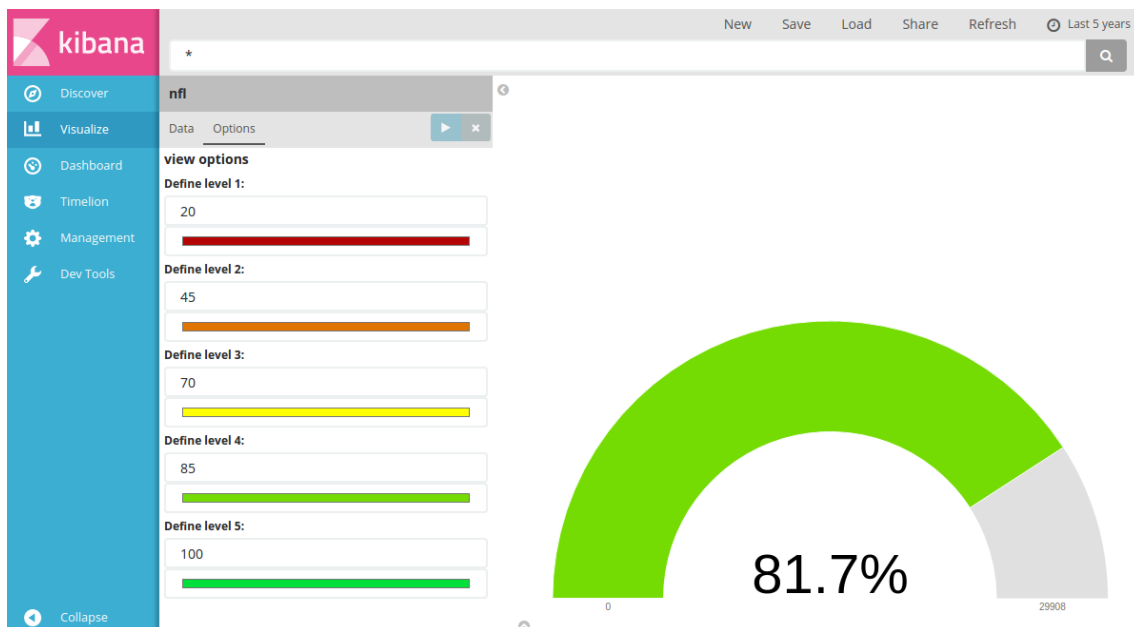


Figura 3.10: Medidor radial en Kibana 5

Una vez implementado el reloj digital había llegado el momento de integrar la biblioteca gráfica que habíamos elegido para empezar a trabajar con ella. Sin embargo justo entonces el equipo de Elastic lanzó una nueva versión de Kibana, la número 5. Lo más lógico era migrar hacía la nueva versión ya que es la que van a empezar a utilizar los usuarios. El primer paso necesario fue descargar la nueva versión de Kibana y revisar su código ya que habían cambiado varios aspectos acerca de la implementación de *plugins*. Una vez conseguido adaptar el reloj

¹<https://docs.angularjs.org/api/ng/filter/date>

digital a la nueva versión de Kibana, se comenzó a trabajar con las visualizaciones ofrecidas por la biblioteca C3.js.

La visualización más sencilla del paquete de C3.js es el medidor radial (*gauge chart*). Un gráfico de medidor radial tiene un arco circular y muestra un único valor que mide el progreso hacia un objetivo o KPI. El progreso hacia ese objetivo se representa mediante el sombreado. Todos los valores posibles están repartidos por igual a lo largo del arco, del mínimo (valor más a la izquierda) al máximo (valor más a la derecha). Como funcionalidad añadida se permite al usuario definir 5 niveles a los que asignar diferentes colores (figura 3.10). Dicha representación fue el primer paso para desarrollar gráficas más complejas que veremos en el siguiente ciclo.

3.6. Iteración 4. Implementación del plugin

3.6.1. Objetivos

Esta es la iteración clave del proyecto ya que en esta fase se construye la aplicación que cumple los requisitos del proyecto, es decir, un *plugin* que funcione con Kibana 5 utilizando la base de datos Elasticsearch y que se apoye en la biblioteca gráfica C3.js. Durante la implementación de la aplicación se han integrado visualizaciones de área, barras, líneas y se han implementado algunas mejoras. Los objetivos de la última iteración son los siguientes:

1. Añadir nuevas visualizaciones
2. Customización e integración en un *dashboard*
3. Dar visibilidad a la herramienta y crear un sitio web

3.6.2. Nuevas visualizaciones

Después de haber testado el funcionamiento del medidor radial me dispongo a añadir nuevas visualizaciones, más complejas y que tengan mayor utilidad. Las visualizaciones que tienen un comportamiento parecido son las de área, líneas y barras que permiten crear un panel de

control común que se utilice para modificar los parámetros de todas ellas. Otros gráficos, implementados en C3.js, como por ejemplo diagramas de dispersión (*scatter plot*) o gráficas de tarta (*pie chart*) se quedan en un segundo plano debido a que tienen un comportamiento diferente y lo más lógico es crear un *plugin* individual para cada visualización.

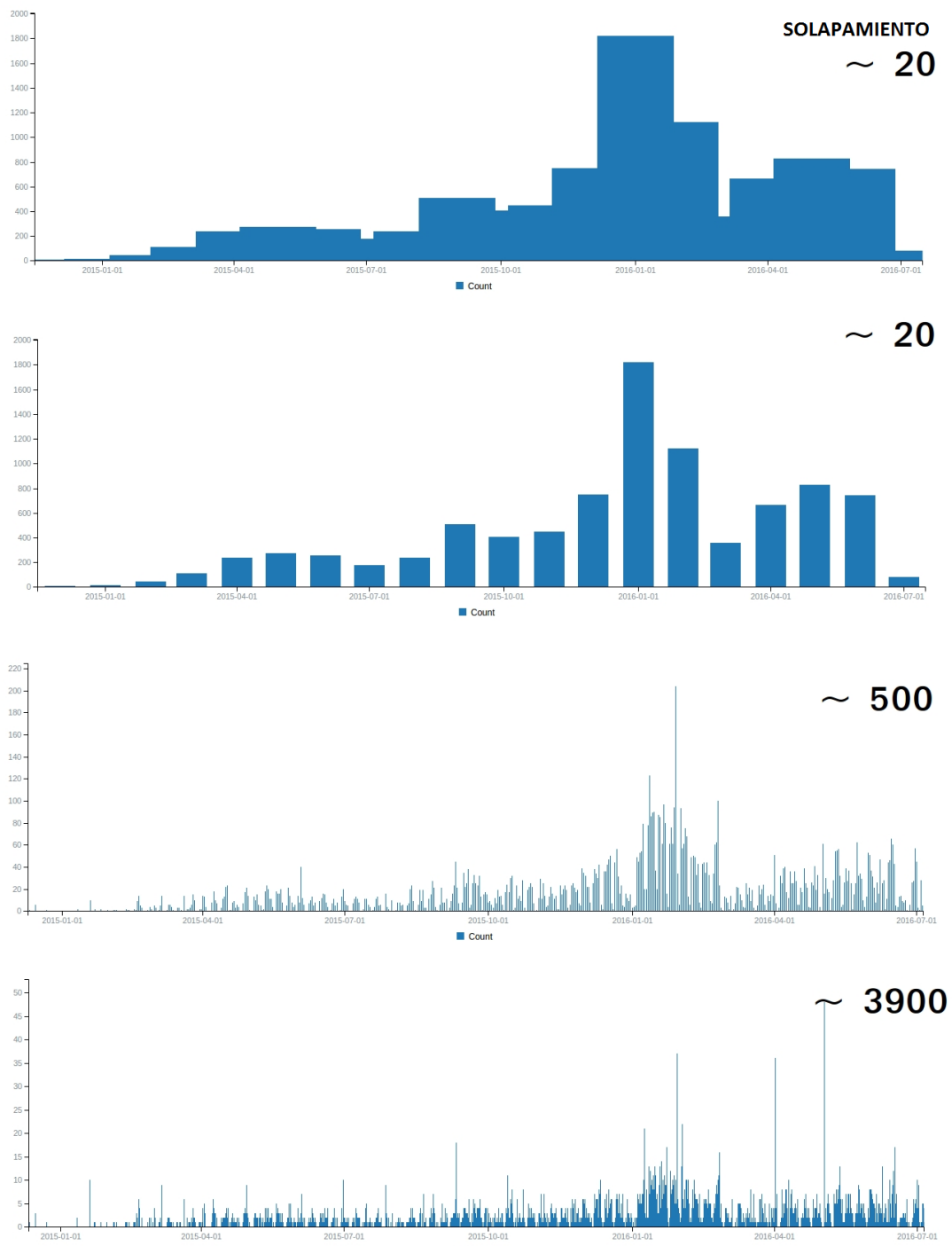


Figura 3.11: Ajuste anchura del gráfico de barras

Uno de los principales problemas encontrados a la hora de integrar la biblioteca de visualización fue con la gráfica de barras ya que cuando se representan series temporales se producía un solape entre las barras. Dicho fallo pudo resolverse, gracias a la API de C3.js, multiplicando el factor de anchura de las barras por una variable en función de la cantidad de datos que se desea representar, a medida que crecen los datos las barras se hacen más estrechas evitando el solapamiento. También se tuvieron que ajustar los valores del eje de ordenadas para que cuando no hubiese valores negativos el valor mínimo sea cero.

3.6.3. Especificaciones añadidas

Una vez comprobado que los datos se representan correctamente se fueron introduciendo otro tipo de mejoras, implementadas en la biblioteca JavaScript, algunas de las cuales no están disponibles en las visualizaciones de Kibana por defecto. Dichas mejoras son las siguientes:

- Personalización del tipo de visualización y del color.
- Combinar varias visualizaciones en el mismo *canvas*.
- Posibilidad de hacer zoom.
- Modificación de la posición de la leyenda (parte inferior o derecha).
- Asignación de etiquetas.
- Reducir la cantidad de datos mostrados en el eje de abscisas cuando se representan datos categóricos para evitar la saturación.
- Etiquetar el eje de abscisas.
- Mostrar el rango temporal que se está representando en series temporales.
- Apilar visualizaciones de tipo barras (*stacked bar mode*).
- Mostrar los valores de todas las visualizaciones a la vez en cada punto de la gráfica.
- Cuadrícula.
- Enfocar datos al colocar el ratón por encima de una etiqueta.

- Descartar o recargar datos cuando la representación haya sido renderizada.
- Integrar las visualizaciones en los cuadros de mando (*dashboard*).

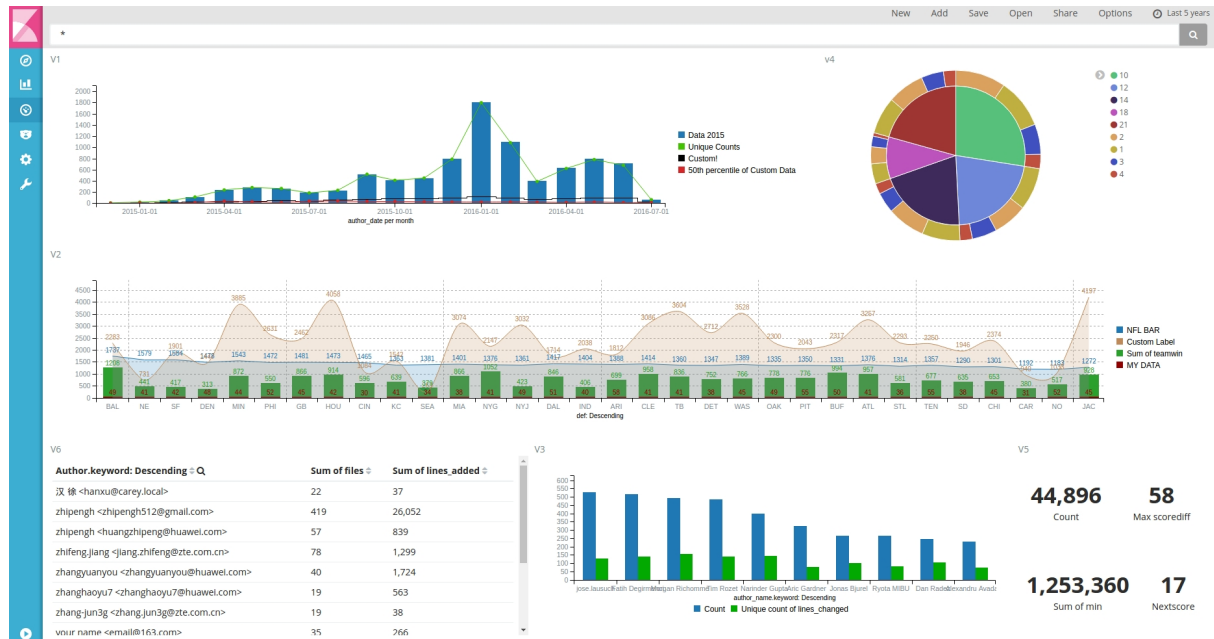


Figura 3.12: Dashboard Kibana 5

3.6.4. Publicación del plugin y sitio web

Como ya se ha mencionado en el apartado 1.5, este software se encuentra alojado en GitHub desde el comienzo de su desarrollo. Posteriormente fue creada una página web oficial, también disponible en GitHub, con el objetivo de ofrecer una imagen más profesional de cara a los usuarios y hacer publicidad de la aplicación. Para la creación del sitio web se ha utilizado Bootstrap con la intención de aprovechar sus principales características como por ejemplo el diseño responsivo, la flexibilidad, las animaciones o el carrousel.

También se ha tratado de dar soporte a las personas que han deseado utilizar o probar esta aplicación y se han contestado numerosos correos respondiendo dudas o recibiendo ideas (y quejas) sobre posibles mejoras e implementaciones en un futuro. Por otro lado, algunos usuarios han preferido publicar comentarios directamente en el repositorio de GitHub que gratamente han sido contestados. La idea es seguir desarrollando el *plugin*, adaptarlo a las futuras versiones de Kibana y traer nuevas funcionalidades.

Capítulo 4

Resultados

4.1. Arquitectura del plugin

En esta sección me gustaría tomarme un momento para explicar la arquitectura de la aplicación y el cometido de cada archivo. Kibana es muy estricta con la organización de los archivos y por lo tanto si queremos que la extensión creada funcione es necesario que el código esté localizado dentro de la carpeta de *plugins* correspondiente y tenga una estructura determinada. De esta forma al arrancar el servicio, Kibana se dará cuenta de que existe una nueva aplicación y la añadirá a la lista de visualizaciones ofrecidas. La única restricción impuesta es que el nombre del *plugin* debe comenzar por una letra minúscula.

- **package.json:** en este archivo, que debe estar localizado en la raíz del proyecto, quedan reflejados los metadatos relevantes (nombre del proyecto, versión, autor, etc.). Este JSON también proporciona información a NPM y permite manejar las dependencias.
- **index.js:** este archivo JavaScript sirve para registrar una nueva aplicación en Kibana. Dentro del objeto `uiExports` se define el tipo de aplicación que en este caso es de tipo visualización (`VisType`) y la carpeta en la que se encuentran alojados los archivos.
- **/node_modules/:** esta carpeta contiene los archivos de los paquetes que hayan sido declarados en el archivo `package.json` o sean dependencias indirectas.
- **/public/:** es la carpeta que contiene todos los archivos de la aplicación, tanto de la lógica

de negocio (controller) como de visualización (HTML y CSS).

- **/public/c3_charts_vis.js:** es el archivo principal del *plugin* en el que se importan el resto de archivos necesarios para el funcionamiento de la aplicación, se definen los parámetros por defecto y los esquemas (métricas y agregaciones).
- **/public/c3_charts_vis.html:** define la plantilla HTML que se va a utilizar para mostrar las visualizaciones dentro de una etiqueta `div`.
- **/public/c3_charts_vis.less:** define el estilo de las visualizaciones a través de variables CSS.
- **/public/c3_charts_vis_controller.js:** contiene toda la lógica de negocio de la aplicación, la forma de obtener los datos de Elasticsearch, modificación de parámetros, composición y representación de las gráficas.
- **/public/c3_charts_vis_params.html:** contiene el código HTML que define el panel de control (parte izquierda de la aplicación) en el que el usuario puede modificar las agregaciones de búsqueda y los parámetros de personalización de las representaciones.

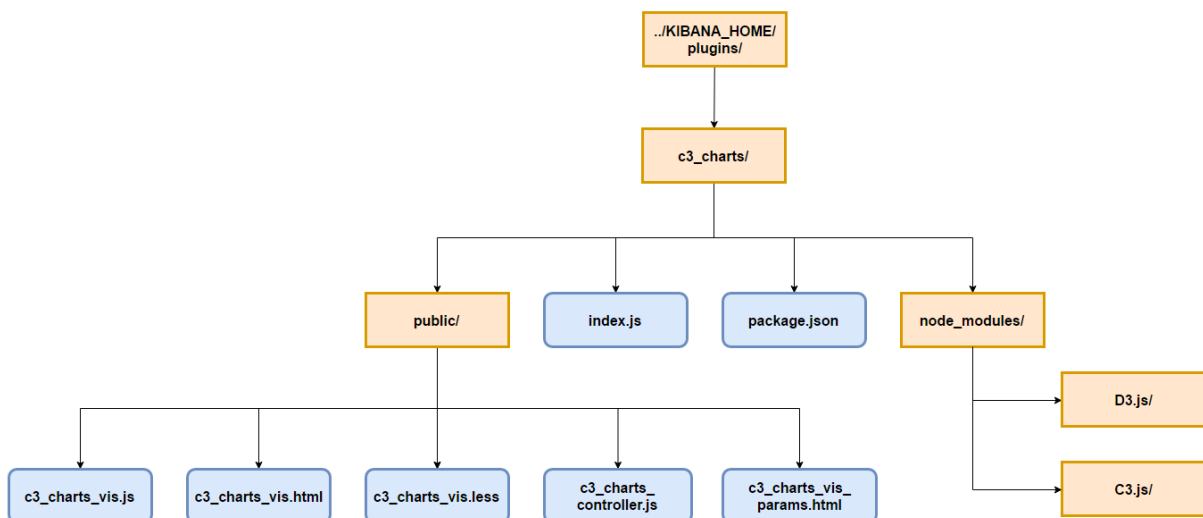


Figura 4.1: Diagrama de archivos

4.2. Funcionamiento del widget

Para comenzar a utilizar la aplicación es necesario seguir los pasos detallados en el repositorio de Github¹ y tener previamente instalados las versiones correspondientes de Kibana y Elasticsearch.

Dentro de la pestaña “visualizar” en la parte izquierda se encuentra la lista de los diferentes *widgets* disponibles, ordenados alfabéticamente, y en la parte derecha se sitúan las visualizaciones guardadas previamente. Cada tipo de representación gráfica tiene un título, una breve descripción de su funcionamiento y un icono como se muestra en la figura inferior. Una vez seleccionada una visualización será necesario indicar el índice de la base de datos que se desea utilizar.

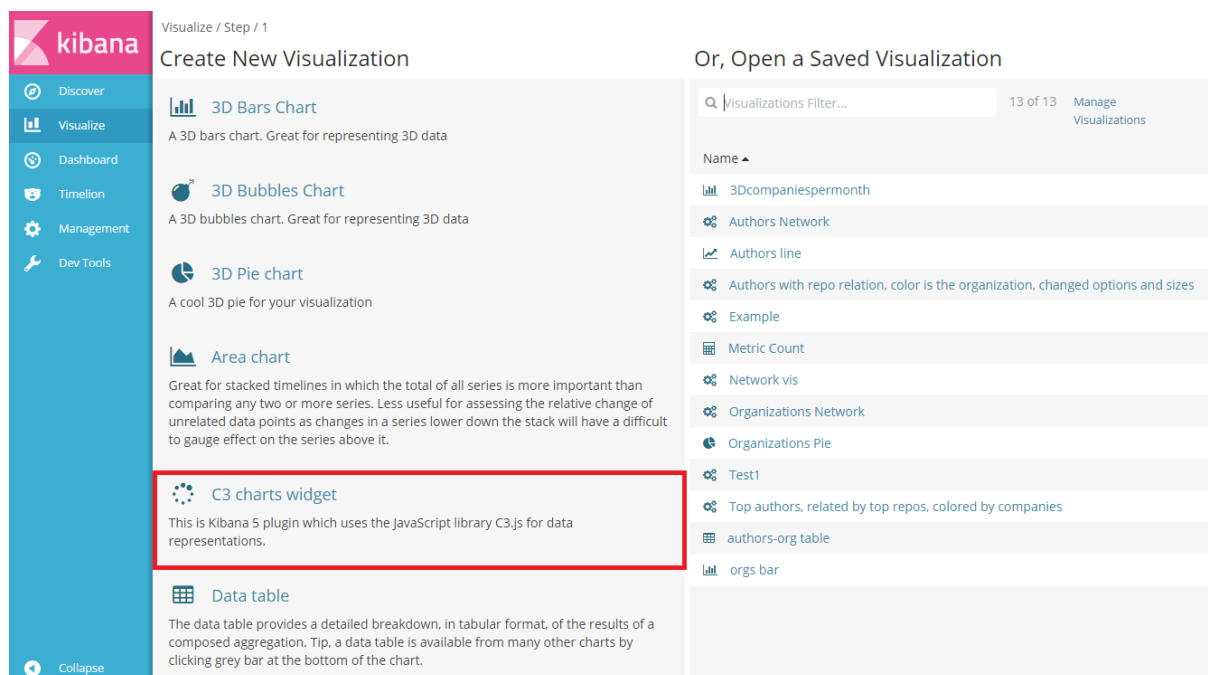


Figura 4.2: Lista de visualizaciones

Una vez completados los pasos anteriores podemos comenzar a utilizar el *plugin*. La apariencia que presenta es muy sencilla y puede dividirse en dos partes. La parte derecha, bordeada de color rojo en la figura inferior, es la ventana de visualización en la que aparecen las representaciones gráficas una vez que hayan sido renderizadas. La parte izquierda, señalada con un

¹https://github.com/mstoyano/kbn_c3js_vis

borde de color azul, es lo que se denomina el panel de control. Dicho panel tiene dos pestañas, la parte de datos que es genérica para todas las aplicaciones (aunque se permite cierta modificación ya que no todos los *widjets* necesitan los mismos recursos) y la parte de las opciones que es propia de cada *plugin*.

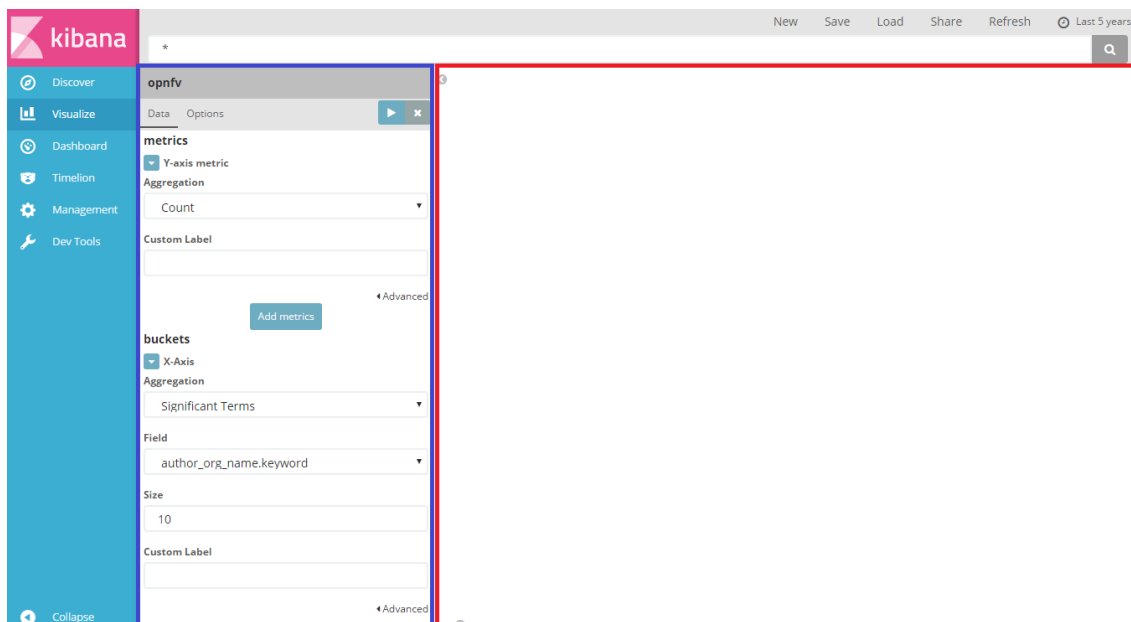


Figura 4.3: Apariencia de la aplicación

En la pestaña de los datos el usuario selecciona y define las diferentes agregaciones y métricas que quiere aplicar, además de la posibilidad de poner etiquetas. Las agregaciones crean un número determinado de *buckets* en los que se distribuyen los datos. Por su parte las métricas son funciones matemáticas que se aplican a los datos de cada *bucket*. En la pestaña de opciones se encuentra toda la funcionalidad añadida para que el usuario pueda customizar sus representaciones.

En la parte superior del panel de opciones se modifica el tipo de gráfica (de área, escalonada, barras o líneas) y el color de cada representación. El menú va aumentando o disminuyendo de forma dinámica en función del número de métricas que tengamos seleccionadas hasta un máximo de 5. Por defecto, cuando se genera una nueva gráfica siempre es de de tipo línea y se muestra con colores predefinidos. En la parte inferior de las opciones se localizan varias casillas de confirmación (*checkbox*):

- **Enable zoom:** si se activa la casilla se puede hacer zoom y desplazar la gráfica utilizando

el cursor. Esto permite poner énfasis en ciertas regiones.

- **Hide points:** esta opción se utiliza con las visualizaciones de área o líneas ocultando los puntos y haciendo que la gráfica sea más suave.
- **Activate data labels:** se muestran los valores representados en cada punto de la gráfica, es ideal para combinar con la opción anterior para evitar la utilización de la ventana de datos emergente, *tooltip*.
- **Show grid lines:** esta opción sirve para activar la cuadrícula.
- **Stacked bar mode:** funciona con dos o más representaciones de barras que por defecto se colocan una al lado de la otra (*grouped*). Con esta opción es posible apilar todas las barras en una única por cada punto de la gráfica.
- **Show just a few x-axis tick values:** cuando se representa un gran volumen de datos, el eje de abscisas puede quedar saturado por la información mostrada. Con esta opción se mostrarán solo algunas etiquetas para que no se rompa la estética de la visualización.
- **Choose legend position:** este desplegable permite seleccionar la posición de la leyenda que se encuentra a la derecha por defecto pero que también puede situarse en la parte inferior.

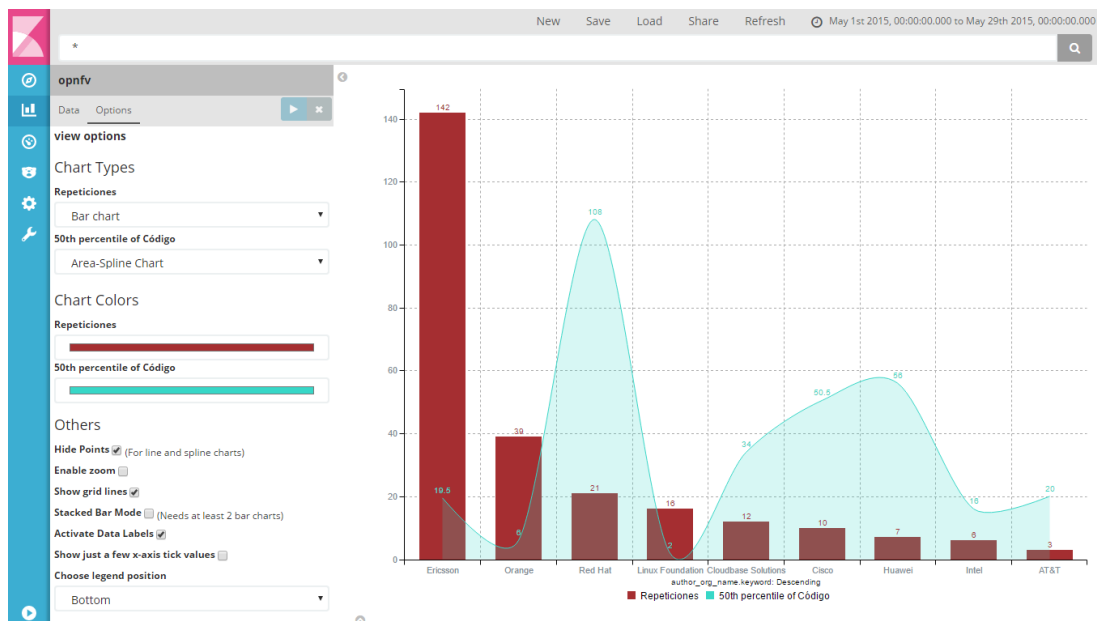


Figura 4.4: Panel de opciones

Además, la biblioteca C3.js ofrece tres características estrella que pueden crear la diferencia respecto a las demás. Dichas particularidades, además de la facilidad de uso y el amplio abanico de customización, hicieron que C3.js sea la biblioteca gráfica elegida como la más adecuada para este proyecto.

- **Tooltip:** el cuadro de descripción emergente resulta muy útil a la hora de reflejar la información ya que muestra los datos de forma conjunta en cada punto de la gráfica. Además, permite modificar su posición al desplazar el cursor verticalmente.

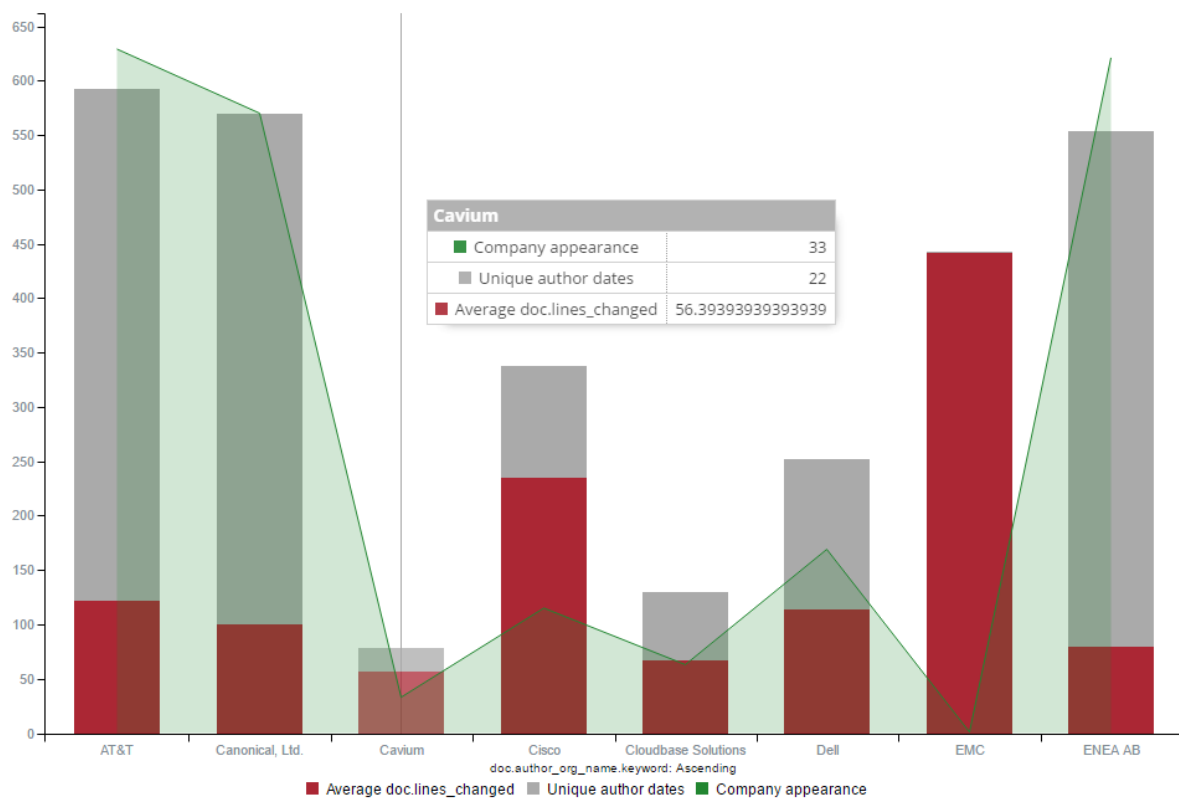


Figura 4.5: Cuadro de descripción

- **Focus:** cuando se representan varios datos en la misma gráfica existe la posibilidad de enfocar una visualización concreta posicionando el cursor por encima de la leyenda. De esta manera el resto de datos quedan en segundo plano.
- **Real time modification:** Gracias a la variedad de APIs, C3.js permite acceder al estado de la visualización y actualizar (descartar o volver a cargar datos) incluso cuando la gráfica

ha sido renderizada. Tan solo es necesario hacer click sobre el título en la leyenda de la gráfica que se desea actualizar.

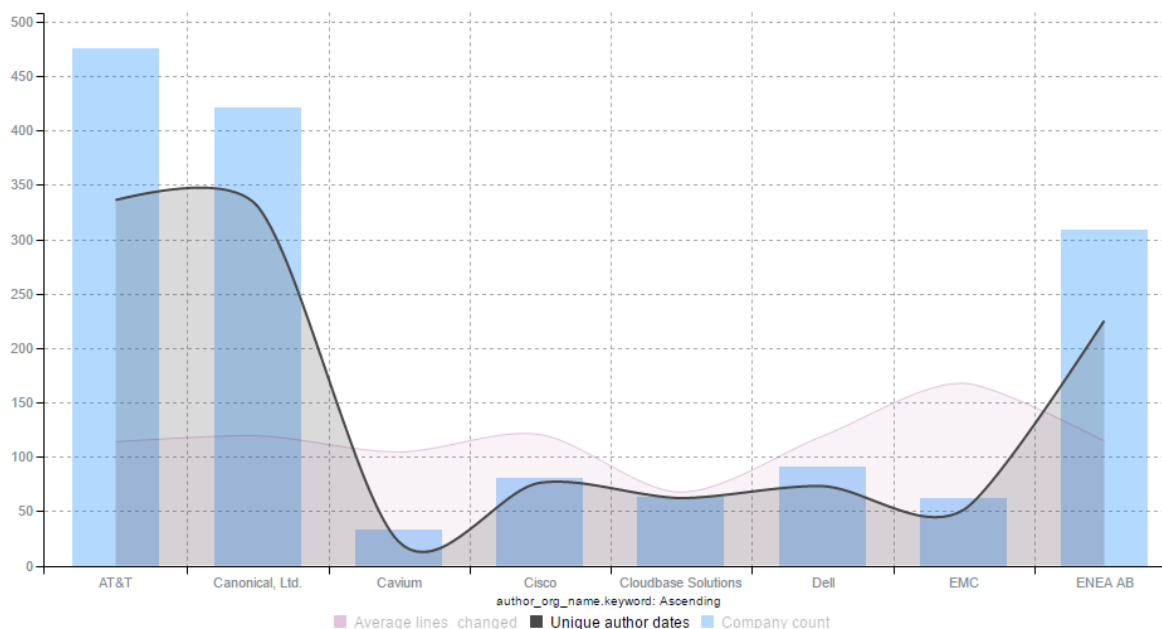


Figura 4.6: Opción de enfoque

Al terminar de representar y customizar una gráfica, Kibana ofrece la posibilidad de guardar el objeto JSON de la visualización (también es posible importar objetos). De esta forma cada vez que se arranque Kibana tendremos acceso al trabajo realizado anteriormente de una forma muy rápida. En la parte derecha de la pestaña de visualización se muestra la lista de visualizaciones guardadas (figura 4.2).

Varias representaciones guardadas pueden ser representadas a la vez para tener mejor visibilidad sobre los datos. Para ello es necesario acceder a la pestaña *dashboard* y seleccionar la opción “añadir” para cargar las visualizaciones deseadas. Es una forma muy práctica de generar un cuadro de mando con diferentes tipos de representaciones gráficas.

Cada visualización es dinámica y permite la alteración del orden y/o tamaño para dar preferencia a ciertos datos. También existe la posibilidad de modificar los parámetros editando la gráfica en la pestaña de “visualización”. Al igual que ocurre con las visualizaciones individuales, tenemos la opción de guardar el *dashboard* generado para que pueda ser utilizado en futuras ocasiones o compartirlo mediante código HTML y/o enlace URL.

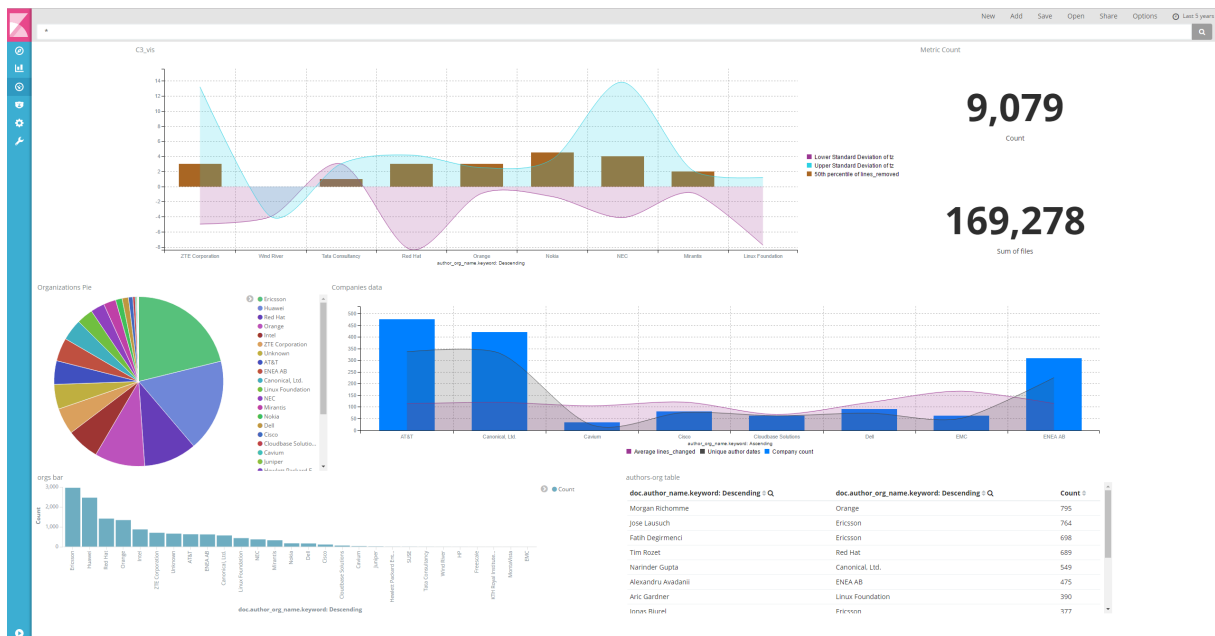


Figura 4.7: Integración de visualizaciones en un cuadro de mando

Capítulo 5

Conclusiones

En este último capítulo llega el momento de echar la vista hacia atrás, hacia los objetivos marcados al comienzo del proyecto con la intención de comprobar si han sido cumplidos satisfactoriamente.

El objetivo principal fue la creación de una aplicación que permita ampliar el rango de posibles representaciones gráficas ofrecidas por el sistema complejo de visualización de datos denominado Kibana. Viendo los resultados obtenidos, de manera individual o gracias a otros usuarios que hayan probado esta aplicación, podemos concluir que el objetivo se ha cumplido satisfactoriamente. Claro está que para poder efectuar el objetivo principal también han tenido que cumplirse los objetivos secundarios.

Para poder desarrollar este *plugin* ha sido necesario, previamente, pasar por una fase de aprendizaje de JavaScript, uno de los lenguajes más utilizados hoy en día. También fue necesario conocer el funcionamiento de las herramientas Kibana y Elasticsearch, que han sido la base del proyecto. Por otro lado, varias bibliotecas JavaScript de visualización de datos han sido estudiadas con el objetivo de elegir la más adecuada para este proyecto. Una vez conseguida la integración de la biblioteca de visualización con Kibana, me he centrado en pulir el funcionamiento de la aplicación paliando con *bugs* que han ido saliendo. Se ha hecho hincapié en añadir funcionalidades implementadas en C3.js para permitir un mayor grado de customización de las visualizaciones a los usuarios. El logro estrella es la integración de las nuevas visualizaciones en un *dashboard* junto con otras representaciones y la respuesta dinámica al cambio de dimensión

de la ventana o el rango de los datos representados.

Sin ninguna duda la parte más difícil del proyecto ha sido aprender a utilizar y entender el funcionamiento de las herramientas de Elastic, Kibana y Elasticsearch. Se trata de *software* relativamente reciente que se desarrolla a ritmos acelerados (personalmente he tenido que migrar a la versión más reciente durante la implementación del proyecto), lo que hace que la información disponible sea escasa y pobre. Fue necesario dedicar mucho tiempo a recabar información, consultar foros, leer código [14] y hacer pruebas para poder entender ciertos conceptos. Sin embargo todo el esfuerzo ha tenido su recompensa ya que mi trabajo ha sido valorado en la comunidad de usuarios de Kibana. El repositorio del proyecto recibe cientos de visitas y tiene decenas de clones (según datos bisemanales de tráfico ofrecidos por GitHub). Tengo el placer de obtener el *feedback* de los usuarios a través de publicaciones en el repositorio o por correo electrónico e intercambiar información acerca de sus proyectos. Además los creadores de Elastic se han tomado la molestia de mencionar esta extensión en su página oficial, algo que me hace sentir muy orgulloso.

5.1. Aplicación de lo aprendido

Durante el desarrollo de este proyecto he podido darme cuenta de la importancia de los conceptos y conocimientos adquiridos a lo largo de la carrera que me han permitido llevarlo a cabo. Las asignaturas que están más relacionadas con la programación y las aplicaciones web son las siguientes:

1. **Fundamentos de la programación:** Fue la asignatura que me enseñó lo que es la programación y los conceptos básicos de cualquier lenguaje (bucles, condiciones, funciones, etc.). Podríamos decir que fue el inicio de una época de mi vida.
2. **Arquitectura de Redes de Ordenadores y Sistemas Telemáticos:** Aunque no se trata de asignaturas enfocadas a la programación sí que he aprendido conceptos importantes como por ejemplo la arquitectura OSI, TCP/IP y varios protocolos.
3. **Programación de Sistemas de Telecomunicación:** Aprendí a programar en *Ada* a través

de las técnicas de programación de aplicaciones telemáticas, niveles de transporte y jerarquía de protocolos.

4. **Software de Sistemas:** Gracias a la cual aprendí el lenguaje C y el funcionamiento de los sistemas operativos modernos.
5. **Servicios y Aplicaciones en Redes de Ordenadores:** Sin duda la más importante para la realización de este TFG, ya que aprendí a programar páginas web con *Python* y *Django*. Me familiaricé con HTML y CSS, además del modelo cliente - servidor.

5.2. Lecciones aprendidas

Gracias a este proyecto he podido reforzar algunos conceptos aprendidos durante la carrera y adquirir otros nuevos.

- Aprender el lenguaje JavaScript y manejar el *framework* AngularJS.
- Reforzar conceptos aprendidos sobre maquetación HTML, CSS y Bootstrap.
- Utilizar y desarrollar la herramienta *open - source* de visualización de datos Kibana.
- Manejar la base de datos NoSQL Elasticsearch.
- Dar a conocer mi aplicación en GitHub, dar soporte y conversar con otros desarrolladores.
- Aprender a utilizar \LaTeX para escribir la memoria del proyecto.

5.3. Trabajos futuros

Dado que ningún software se termina, este proyecto puede seguir desarrollándose en el futuro. Las posibles mejoras que podrían introducirse son las siguientes:

- Lanzar actualizaciones con las nuevas versiones de Kibana para asegurar la compatibilidad del *plugin*.

- Implementar nuevas funcionalidades como por ejemplo añadir un segundo eje de ordenadas, crear regiones, representar sumas aditivas, habilitar varios niveles de *buckets*, etc. para permitir una mejor personalización a los usuarios.
- Crear varios niveles de *buckets* para que el usuario pueda filtrar la información de forma más específica y concreta.
- Incorporar nuevas visualizaciones (tarta, donut, medidor radial y gráfico de dispersión (*scatter plot*)) para completar el conjunto de gráficas ofrecido por C3.js.
- Crear nuevos módulos de visualización de datos.

Apéndice A

Código de la aplicación

A.1. index.js

```
export default function (kibana) {  
  
  return new kibana.Plugin({  
    uiExports: {  
      visTypes: [  
        'plugins/k5p-c3/c3_vis'  
      ]  
    }  
  });  
};
```

A.2. package.json

```
{  
  "name": "k5p-c3",  
  "description": "This is Kibana 5 plugin which uses the JavaScript  
    library C3.js for data representations",
```

```
"version": "5.0.0-rc1",
"homepage": "https://mstoyano.github.io/",
"license": "Apache-2.0",
"author": "Momchil Stoyanov <mom4il13@hotmail.com>",
"repository": {
  "type": "git",
  "url": "https://github.com/mstoyano/kbn_c3js_vis"
},
"dependencies" : {
  "c3": "0.4.11"
}
}
```

A.3. c3_charts_vis.html

```
<div ng-controller="KbnC3VisController" class="c3-vis">
  <div class="c3-container">
    <div class="chartc3"></div>
    <div class="text-center" ng-hide="$root.show_chart"><h4>{{
      waiting }}</h4></div>
  </div>
</div>
```

A.4. c3_charts_vis.js

```
import 'plugins/k5p-c3/c3_vis.less';
import 'plugins/k5p-c3/c3_vis_controller';
import TemplateVisTypeTemplateVisTypeProvider from 'ui/
  template_vis_type/template_vis_type';
import VisSchemasProvider from 'ui/vis/schemas';
```

```
import c3VisTemplate from 'plugins/k5p-c3/c3_vis.html';
import c3VisParamsTemplate from 'plugins/k5p-c3/c3_vis_params.html';

// register the provider with the visTypes registry
require('ui/registry/vis_types').register(c3VisProvider);

// Require the JavaScript CSS file
require('../node_modules/c3/c3.css');

function c3VisProvider(Private) {
  const TemplateVisType = Private(
    TemplateVisTypeTemplateVisTypeProvider);
  const Schemas = Private(VisSchemasProvider);

  return new TemplateVisType({
    name: 'c3Charts',
    title: 'C3 charts widget',
    icon: 'fa-spinner',
    description: 'This is Kibana 5 plugin which uses the
      JavaScript library C3.js for data representations',
    template: c3VisTemplate,
    params: {
      defaults: {
        type1: 'line',
        color1: '#1f77b4',
        type2: 'line',
        color2: '#ff7f0e',
        type3: 'line',
        color3: '#2ca02c',
        type4: 'line',
        color4: '#d62728',
        type5: 'line',
        color5: '#9467bd',
```

```
    enableZoom: false,  
    dataLabels: false,  
    hidePoints: false,  
    gridlines: false,  
    few_x_axis: false,  
    legend_position: "right",  
    time_format: "%d-%m-%Y",  
    grouped: false  
  
    },  
  
    editor: c3VisParamsTemplate  
    },  
  
    schemas: new Schemas([  
    {  
        group: 'metrics',  
        name: 'metric',  
        title: 'Y-axis metric',  
        min: 1,  
        max: 5,  
        defaults: [ { type: 'count', schema: 'metric' } ],  
    },  
    {  
        group: 'buckets',  
        name: 'buckets',  
        title: 'X-Axis',  
        min: 1,  
        max: 1,  
        aggFilter: ['!geohash_grid']  
    }])  
    });  
}
```



```
// export the provider so that the visType can be required with
  Private()
export default c3VisProvider;
```

A.5. c3_charts_vis.less

```
.c3-vis {
  width: 100%;
  display: flex;
  flex-direction: row;
  flex-wrap: wrap;
  justify-content: space-around;
  align-items: center;
  align-content: space-around;

  .c3-container {
    text-align: center;
    padding: 1em;
    width: 100%;
    position: relative;
  }
}
```

A.6. c3_charts_vis_controller.js

```
import uiModules from 'ui/modules';
import AggResponseTabifyTabifyProvider from 'ui/agg_response/tabify/
  tabify';
```

```
import errors from 'ui/errors';

// get the kibana/table_vis module, and make sure that it requires
// the "kibana" module if it didn't already
const module = uiModules.get('kibana/c3_vis', ['kibana']);

// Require C3.js
const c3 = require('c3');

module.controller('KbnC3VisController', function($scope, $element,
  Private){

  var hold = "";
  var wold= "";
  $scope.$root.label_keys = [];
  $scope.$root.editorParams = {};
  $scope.$root.activate_grouped = false;
  const tabifyAggResponse = Private(AggResponseTabifyTabifyProvider);
  var x_axis_values = [];
  var timeseries = [];
  var parsed_data = [];
  var chart_labels = {};
  var x_label = "";
  var time_format = "";

  // Identify the div element in the HTML
  var idchart = $element.children().find(".chartc3");
  const message = 'This chart require more than one data point. Try
    adding an X-Axis Aggregation.';

  // Be alert to changes in vis_params
  $scope.$watch('vis.params', function (params) {
```

```
    if (!$scope.$root.show_chart) return;
    //if (Object.keys(params.editorPanel).length == 0 && params.
        enableZoom == previo_zoom) return;
    $scope.chartGen();
});

// C3JS chart generator
$scope.chart = null;
$scope.chartGen = function() {

    // change bool value
    $scope.$root.show_chart = true;

    //create data_colors object
    var the_labels = Object.keys(chart_labels);
    var data_colors = {};
    var data_types = {};
    var i = 0;
    var create_color_object = the_labels.map(function(chart) {
        if (i == 0) {
            data_colors[chart] = $scope.vis.params.color1;
            data_types[chart] = $scope.vis.params.type1;

        } else if (i == 1) {

            data_colors[chart] = $scope.vis.params.color2;
            data_types[chart] = $scope.vis.params.type2;

        } else if (i == 2) {
            data_colors[chart] = $scope.vis.params.color3;
            data_types[chart] = $scope.vis.params.type3;
```

```
    } else if (i == 3){
      data_colors[chart] = $scope.vis.params.color4;
      data_types[chart] = $scope.vis.params.type4;

    } else if (i == 4){
      data_colors[chart] = $scope.vis.params.color5;
      data_types[chart] = $scope.vis.params.type5;
    }

    i++;

  });

  // count bar charts and change bar ratio
  var the_types = Object.values(data_types);
  var chart_count = {};
  the_types.forEach(function(i){ chart_count[i] = (chart_count[i]
    || 0)+1; });

  if (chart_count.bar){

    var my_ratio = 5 / timeseries.length;
    my_ratio = (my_ratio > 0.35) ? my_ratio = 0.3 : my_ratio;

    if (chart_count.bar > 1){

      my_ratio = (my_ratio < 0.02) ? my_ratio = 0.02 : my_ratio;
      $scope.$root.activate_grouped = true;

    } else {

      my_ratio = (my_ratio < 0.01) ? my_ratio = 0.01 : my_ratio;
```

```
    $scope.$root.activate_grouped = false;
  }

}

var bucket_type = $scope.vis.aggs.bySchemaName['buckets'][0].type
  .name;

// define the data to representate
if (parsed_data.length == 1) {
  var total_data = {'x': 'x1', 'columns': [timeseries,
    parsed_data[0]]};
} else if (parsed_data.length == 2) {
  var total_data = {'x': 'x1', 'columns': [timeseries,
    parsed_data[0], parsed_data[1]]};
} else if (parsed_data.length == 3) {
  var total_data = {'x': 'x1', 'columns': [timeseries,
    parsed_data[0], parsed_data[1], parsed_data[2]]};
} else if (parsed_data.length == 4) {
  var total_data = {'x': 'x1', 'columns': [timeseries,
    parsed_data[0], parsed_data[1], parsed_data[2], parsed_data
    [3]]};
} else {
  var total_data = {'x': 'x1', 'columns': [timeseries,
    parsed_data[0], parsed_data[1], parsed_data[2], parsed_data
    [3], parsed_data[4]]};
}

// largest number possible in JavaScript.
var global_min = Number.MAX_VALUE;

// Search the min value of the data
var parsed_data_copy = JSON.parse(JSON.stringify(parsed_data));
```

```
var cada_array = parsed_data_copy.map(function(each_array){

    each_array.splice(0, 1);
    // ECMAScript 6 spread operator
    var each_array_min = Math.min(...each_array);
    global_min = (each_array_min < global_min) ? each_array_min :
        global_min;

});

global_min = (global_min >= 0) ? 0 : global_min;

// configurate C3 object
var config = {};
config.bindto = idchart[0];
config.data = total_data;
config.data.types = data_types;
config.data.colors = data_colors;
config.data.labels = $scope.vis.params.dataLabels;
config.legend = {"position": $scope.vis.params.legend_position};

// timeseries config
if (bucket_type == "date_histogram" || bucket_type == "date_range
    ") {

    config.bar = {"width": {"ratio": my_ratio}};

    var last_timestapm = timeseries[timeseries.length-1];
    var first_timestamp = timeseries[1];
    var timestamp_diff = last_timestapm - first_timestamp;

    // Time format
```

```
    if (timestamp_diff > 86400000){
      time_format = "%Y-%m-%d";
    } else {
      time_format = "%H:%M";
    }

    var bool_fit = false;
    bool_fit = (timeseries.length < 4) ? bool_fit = true : bool_fit
      = false;

    config.axis = {"x": {"label": {"text": x_label, "position": '
      outer-center'}, "type": "timeseries", "tick": {"fit":
      bool_fit, "multiline": false, "format": time_format}}, "y": {
      "min": global_min, "padding": {"top": 30, "bottom": 0 }}};
    config.tooltip = {"format": {"title": function (x) {return x;
      }};

    if ($scope.vis.params.legend_position == "bottom"){
      config.padding = {"right": 20};
    }

    // category data config
  } else {

    config.axis = {"x": {"label": {"text": x_label, "position": '
      outer-center'}, "type": "category", "tick": {"multiline":
      false}}, "y": {"min": global_min, "padding": {"top": 30, "
      bottom": 1 }}};

    if (timeseries.length-1 > 13 && $scope.vis.params.few_x_axis){
      config.axis = {"x": {"label": {"text": x_label, "position": '
      outer-center'}, "type": "category", "tick": {"fit": false,
      "multiline": false, "culling": {"max": 10}}}, "y": {"min":
```

```
        global_min, "padding": {"top": 30, "bottom": 1 }));
    }
}

// Group bar charts, we need 2+ bar charts and checked checkbox
// in params
if ($scope.$root.activate_grouped && $scope.vis.params.grouped){

    var los_keys = Object.keys(data_types);
    var los_values = Object.values(data_types);
    var group_charts = [];
    var i = 0;
    var are_they = los_values.map(function(chart_type){

        if (chart_type == "bar"){
            group_charts.push(los_keys[i]);
        }

        i++;

    });

    config.data.groups = [group_charts];
}

if ($scope.vis.params.gridlines){
    config.grid = {"x": {"show": true}, "y": {"show": true}};
}

// zoom and hide points features
config.point = {"show": !$scope.vis.params.hidePoints};
config.zoom = {"enabled" : $scope.vis.params.enableZoom};
```



```
// Generate and draw
$scope.chart = c3.generate(config);

// resize
var elem = $(idchart[0]).closest('div.visualize-chart');
var h = elem.height();
var w = elem.width();
$scope.chart.resize({height: h - 50, width: w - 50});

};

// Get data from ES
$scope.processTableGroups = function (tableGroups) {
  tableGroups.tables.forEach(function (table) {
    table.columns.forEach(function (column, i) {

      var data = table.rows;
      var tmp = [];

      for (var val in data){
        tmp.push(data[val][i]);
      }

      if (i > 0){

        $scope.$root.label_keys.push(column.title);
        chart_labels[column.title] = column.title;
        tmp.splice(0, 0, column.title);
        parsed_data.push(tmp);

      } else {
```

```
        x_label = column.title;
        x_axis_values.push(tmp);
    }
});
});

$scope.$root.editorParams.label = chart_labels;
};

$scope.$watch('esResponse', function(resp) {
    if (resp) {

        if (!$scope.vis.aggs.bySchemaName['buckets']) {
            $scope.waiting = message;
            return;
        }

        x_axis_values.length = 0;
        timeseries.length = 0;
        parsed_data.length = 0;
        chart_labels = {};
        $scope.$root.label_keys = [];
        $scope.processTableGroups(tabifyAggResponse($scope.vis, resp));

        // avoid reference between arrays!!!
        timeseries = x_axis_values[0].slice();
        timeseries.splice(0,0,'x1');
        $scope.chartGen();
    }
});
```

```
// Automatic resizing of graphics
$scope.$watch(
  function () {
    var elem = $(idchart[0]).closest('div.visualize-chart');
    var h = elem.height();
    var w = elem.width();

    if (!$scope.chart) return;

    if (idchart.length > 0 && h > 0 && w > 0) {

      if (hold !== h || wold !== w) {
        $scope.chart.resize({height: h - 50, width: w - 50});
        hold = elem.height();
        wold = elem.width();
      }

    }
  },
  true
);
});
```

A.7. c3_charts_vis_params.html

```
<div class="form-group">
  <form>
    <h4>Chart Types</h4>
    <div class="opt-group">
```

```
<div ng-show="$root.label_keys.length > 0">
  <div class="form-group">
    <label for="sell">{{ $root.label_keys[0] }}</label>
    <select class="form-control" id="sell" ng-model="vis.params
      .type1">
      <option value="bar">Bar chart</option>
      <option value="line">Line Chart</option>
      <option value="spline">Spline Chart</option>
      <option value="step">Step Chart</option>
      <option value="scatter">Dot Chart</option>
      <option value="area">Area Chart</option>
      <option value="area-spline">Area-Spline Chart</option>
      <option value="area-step">Area-Step Chart</option>
    </select>
  </div>
</div>

<div ng-show="$root.label_keys.length > 1">
  <div class="form-group">
    <label for="sell">{{ $root.label_keys[1] }}</label>
    <select class="form-control" id="sell" ng-model="vis.params
      .type2">
      <option value="bar">Bar chart</option>
      <option value="line">Line Chart</option>
      <option value="spline">Spline Chart</option>
      <option value="step">Step Chart</option>
      <option value="scatter">Dot Chart</option>
      <option value="area">Area Chart</option>
      <option value="area-spline">Area-Spline Chart</option>
      <option value="area-step">Area-Step Chart</option>
    </select>
  </div>
</div>
```

```
<div ng-show="$root.label_keys.length > 2">
  <div class="form-group">
    <label for="sell">{{ $root.label_keys[2] }}</label>
    <select class="form-control" id="sell" ng-model="vis.params
      .type3">
      <option value="bar">Bar chart</option>
      <option value="line">Line Chart</option>
      <option value="spline">Spline Chart</option>
      <option value="step">Step Chart</option>
      <option value="scatter">Dot Chart</option>
      <option value="area">Area Chart</option>
      <option value="area-spline">Area-Spline Chart</option>
      <option value="area-step">Area-Step Chart</option>
    </select>
  </div>
</div>

<div ng-show="$root.label_keys.length > 3">
  <div class="form-group">
    <label for="sell">{{ $root.label_keys[3] }}</label>
    <select class="form-control" id="sell" ng-model="vis.params
      .type4">
      <option value="bar">Bar chart</option>
      <option value="line">Line Chart</option>
      <option value="spline">Spline Chart</option>
      <option value="step">Step Chart</option>
      <option value="scatter">Dot Chart</option>
      <option value="area">Area Chart</option>
      <option value="area-spline">Area-Spline Chart</option>
      <option value="area-step">Area-Step Chart</option>
    </select>
  </div>
</div>
```

```

</div>

<div ng-show="$root.label_keys.length > 4">
  <div class="form-group">
    <label for="sel1">{{ $root.label_keys[4] }}</label>
    <select class="form-control" id="sel1" ng-model="vis.params
      .type5">
      <option value="bar">Bar chart</option>
      <option value="line">Line Chart</option>
      <option value="spline">Spline Chart</option>
      <option value="step">Step Chart</option>
      <option value="scatter">Dot Chart</option>
      <option value="area">Area Chart</option>
      <option value="area-spline">Area-Spline Chart</option>
      <option value="area-step">Area-Step Chart</option>
    </select>
  </div>
</div>

</div>
</form>
</div>

<div class="form-group">
  <form>
    <h4>Chart Colors</h4>

    <div ng-show="$root.label_keys.length > 0">
      <label for="sel3">{{ $root.label_keys[0] }}</label>
      <input type="color" name="sel3" id="sel3" ng-model="vis.params.
        color1" class="form-control">
    </div>
  </form>
</div>

```

```
<div ng-show="$root.label_keys.length > 1">
  <label for="sel3">{{$root.label_keys[1]}}</label>
  <input type="color" name="sel3" id="sel3" ng-model="vis.params.
    color2" class="form-control">
</div>

<div ng-show="$root.label_keys.length > 2">
  <label for="sel3">{{$root.label_keys[2]}}</label>
  <input type="color" name="sel3" id="sel3" ng-model="vis.params.
    color3" class="form-control">
</div>

<div ng-show="$root.label_keys.length > 3">
  <label for="sel3">{{$root.label_keys[3]}}</label>
  <input type="color" name="sel3" id="sel3" ng-model="vis.params.
    color4" class="form-control">
</div>

<div ng-show="$root.label_keys.length > 4">
  <label for="sel3">{{$root.label_keys[4]}}</label>
  <input type="color" name="sel3" id="sel3" ng-model="vis.params.
    color5" class="form-control">
</div>

</form>
</div>

<div class="form-group">

  <form>
    <h4>Others</h4>
    <div clas="show-if-chart" ng-show="$root.show_chart">
```

```
<div class="form-group">
  <label> Hide Points </label>
  <input type="checkbox" ng-model="vis.params.hidePoints"> (For
    line and spline charts)
</div>

<div class="form-group">
  <label> Enable zoom </label>
  <input type="checkbox" ng-model="vis.params.enableZoom">
</div>

<div class="form-group">
  <label> Show grid lines </label>
  <input type="checkbox" ng-model="vis.params.gridlines">
</div>

<div class="form-group">
<!-- ng-show="$root.activate_grouped" -->
  <label> Stacked Bar Mode </label>
  <input type="checkbox" ng-model="vis.params.grouped"> (Needs
    at least 2 bar charts)
</div>

<div class="form-group">
  <label> Activate Data Labels </label>
  <input type="checkbox" ng-model="vis.params.dataLabels">
</div>

<div class="form-group">
  <label> Show just a few x-axis tick values </label>
  <input type="checkbox" ng-model="vis.params.few_x_axis">
</div>
```



```
<div class="form-group">
  <label for="sel-legend">Choose legend position</label>
  <select class="form-control" id="sel-legend" ng-model="vis.
    params.legend_position">
    <option value="right">Right</option>
    <option value="bottom">Bottom</option>
  </select>
</div>

</div>
</form>
</div>
```


Apéndice B

Licencia del proyecto

Copyright 2016 - 2017 Momchil Stoyanov

Licensed under the Apache License, Version 2.0 (the “License”); you may not use this file except in compliance with the License. You may obtain a copy of the License at

<http://www.apache.org/licenses/LICENSE-2.0>

Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an “AS IS” BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.

Bibliografía

[1] MARIJN HAVERBEKE *Eloquent JavaScript*. Second Edition - 2011.

<http://eloquentjavascript.net/>

[2] DR. AXEL RAUSCHMAYER *Speaking JavaScript: An In-Depth Guide for Programmers*
O'Reilly Media, Inc - 2014.

<http://speakingjs.com/>

[3] OLEKSII KONONENKO, OLGA BAYSAL, REID HOLMES, AND MICHAEL W. GODFREY
Mining Modern Repositories with Elasticsearch. University of Waterloo, Canada - 2014.

<https://cs.uwaterloo.ca/~okononen/msr2014.pdf>

[4] Elastic: The website of Elasticsearch and Kibana.

<https://www.elastic.co/>

[5] Elastic Stack and Product Documentation - *version 5.1*.

<https://www.elastic.co/guide/index.html>

[6] The official GitHub repository of Kibana.

<https://github.com/elastic/kibana>

[7] C3.js - D3-based reusable chart library.

<http://c3js.org/>

[8] AngularJS: Superheroic JavaScript MVW Framework

<https://angularjs.org/>

[9] Superset: make it easy to slice, dice and visualize data.

<https://github.com/airbnb/superset>

[10] The Jupyter Notebook: a web application that allows you to create and share documents that contain live code, equations, visualizations and explanatory text.

<http://jupyter.org/>

[11] Shaping up with Angular.js, free course.

<https://www.codeschool.com/courses/shaping-up-with-angular-js>

[12] The world's largest web for learning web technologies online - W3school.

<http://www.w3schools.com/angular/>

[13] The Complete Guide to the ELK Stack

<http://logz.io/learn/complete-guide-elk-stack/>

[14] The biggest community of programmers helping us each other.

<http://stackoverflow.com/>