Usabilidad en dashboards para analítica de Big Data

Trabajo Fin de Grado

Curso 2016–2017

Autor
Tomás Muñoz Testón

Director
Manuel Freire

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática

Universidad Complutense de Madrid

Madrid, 2017

Usabilidad en dashboards para analítica de Big Data

Trabajo fin de grado

Tomás Muñoz Testón

Dirigido por

Manuel Freire

Departamento de Ingeniería del Software e Inteligencia Artificial Facultad de Informática

Universidad Complutense de Madrid

Madrid, 2017

Para ti Bea, por hacerme sentir el chico más afortunado
del mundo. Sin ti no habría sido posible.

AGRADECIMIENTOS

A mi tutor Manuel Freire por su gran ayuda, por ser un verdadero profesional en su trabajo y por su cercanía y ganas de enseñar.

A mi profesor Gonzalo Méndez por haberme proporcionado material interesante.

A Dan Cristian por sus horas de dedicación y por ser un gran compañero.

A mis compañeros de facultad Andrea, Eduardo, Iván, Rodrigo, Fernando, Laura, Sergio, Carmen y Samuel por su paciencia, ayuda y motivación en todo momento.

A mi familia, por haber hecho que todo esto sea posible.

A mis padres por ayudarme en todo momento y por hacerme sentir capaz de cualquier cosa.

Y sobre todo a ti Beatriz Pérez Cerrada por tu amor y por hacerme sentir feliz y ayudarme en todo momento. Sin ti no habría sido posible.

AUTORIZACIÓN

Se autoriza a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria como el código, la documentación y/o el prototipo desarrollado.

Tomás Muñoz Testón.

RESUMEN

A día de hoy, una de las mayores industrias en el mundo del entretenimiento es la de los videojuegos. Actualmente, existen multitud de tipos de videojuegos entre los que se encuentran los “juegos serios”. Estos tienen como principal objetivo la educación, cuyo propósito es desafiar y generar metas a los jugadores.

De la iteración entre los usuarios y los juegos serios, se genera información que se puede convertir en relevante y útil gracias a la analítica de aprendizaje.

El presente trabajo fin de grado (a partir de ahora TFG) tiene como objetivo facilitar la creación de nuevas visualizaciones, que ayuden a interpretar los datos y que se integren dentro de una infraestructura de analítica de aprendizaje para su uso en juegos serios.

En este TFG se ha creado una estructura de un plugin de visualización para Kibana, utilizando librerías de visualización externas que facilitan su creación. Para facilitar al máximo la creación de esta estructura, se ha programado un generador de proyectos.

Para comprobar el funcionamiento de toda esta estructura, se realizan varios ejemplos utilizando el generador de plugins, donde se incorporan visualizaciones creadas por el autor de este TFG.

ABSTRACT

Nowadays, one of the biggest industries in the world of entertainment is video games. Currently, there are many types of video games including "serious games". These have as main objective the education, whose purpose is to challenge and generate goals to the players.

From the interaction between the users and the serious games, information is generated and many become relevant and useful thanks to the analytic of learning.

The purpose of this TFG is to facilitate the creation of new visualizations that help to interpret the data and integrate it into a learning analytics infrastructure for the use in serious games.

In this TFG a structure of a visualization plugin for Kibana has been created using external visualizations libraries that facilitate its creation. To facilitate the creation of this structure, a project generator has been programmed.

To verify the operation of this entire structure, several examples have been made using the plugin generator, which incorporates visualizations created by the author of this TFG.

PALABRAS CLAVE

- Analítica de aprendizaje
- Juegos serios
- Big data
- Kibana
- Elasticsearch
- D3.js
- Visualizaciones

KEY WORDS

- Learning analytics
- Serious games
- Big data
- Kibana
- Elasticsearch
- D3.js
- Visualizations

LISTA DE ACRÓNIMOS

- GLA: Game learning analytics.
- GA: Game analytics.
- HTML: HyperText Markup Language
- JSON: Javascriot object notation
- LA: Learning analytics.
- NPM: Node package manager.
- REST: Representational State Transfer
- TFG: Trabajo fin de grado.
- UCM: Universidad Complutense de Madrid.

ÍNDICE

AGRADECIMIENTOS iv

AUTORIZACIÓN v

RESUMEN vi

ABSTRACT vii

PALABRAS CLAVE viii

KEY WORDS ix

LISTA DE ACRÓNIMOS x

Capitulo 1. INTRODUCCIÓN 1

1.1 Introducción 2

1.2 Motivación 4

1.3 Objetivos 6

1.4 Plan de trabajo 6

1.5 Estructura del documento 7

Capitulo 2. Diseño 9

2.1 Visualizaciones en la web 10

Almacenando datos para su visualización 11

Dashboards 12

NodeJS 14

Node Package Manager (NPM) 16

CommonJS 16

Plugins 16

Scaffolding 16

2.2 Diseño elegido 18

2.3 Solución adoptada 18

Kibana y Elasticsearch 18

Plugins para Kibana 20

Capitulo 3. Desarrollo 23

3.1 Proceso de creación de un plugin 24

3.1.1 Diseño de la visualización 25

3.1.2 Requisitos mínimos y creación de un plugin básico 25

3.1.3 Origen y transformación de los datos. Elasticsearch y función proveedora 27

3.2 Ejemplo básico aplicado de un plugin 28

3.2.1 Diseño de la visualización 28

3.2.2 Cargar datos en Elasticsearch 29

3.2.3 Registro de la visualización 29

3.2.4 Definir la visualización 30

3.2.5 Programar el controlador 30

3.2.6 Objeto para visualizar con D3 30

3.3 Ejemplo avanzado de un plugin **31**
3.3.1 Diseño de la visualización 31
3.3.2 Cargar datos en Elasticsearch 31
3.3.3 Registro de la visualización 32
3.3.4 Definir la visualización 32
3.3.5 Programar el controlador 32
3.3.6 Objeto para visualizar con D3 33

Capitulo 4. Resultados **34**
4.1 Plugin-eskid3 ..... **35**
4.2 Generator-plugin-eskid3 ..... **39**
4.3 Plugin-eskid3-exampleBars **41**
4.4 Plugin-eskid3-tree **41**

Capitulo 5. Conclusiones y trabajo futuro **43**
5.1 Conclusiones **44**
5.2 Trabajo futuro **45**

Capitulo 6. Conclusions and future work **46**
6.1 Conclusions **47**
6.2 Future work **48**

Anexo **49**
ANEXO A: Proceso detallado para crear un controlador para Eskid3 ..... **50**
ANEXO B: Requisitos necesarios para crear plugins en Kibana **51**
ANEXO C: Cómo probar un plugin en Kibana **52**

Bibliografía **56**

ÍNDICE DE FIGURAS

Figura 1. Visualizaciones disponibles en Kibana
(https://www.elastic.co/guide/en/kibana/current/images/tutorial-visualize.png) 5

Figura 2. Infraestructura de soporte proporcionada por los componentes de analítica del proyecto RAGE (13) 6

Figura 3. Comparativa Kibana - Grafana- Graphite.
(https://stackshare.io/stackups/graphite-vs-grafana-vs-kibana) 13

Figura 4. Estructura básica de un proyecto NodeJS. 15

Figura 5. Proceso de creación de un plugin para Kibana. 24

Figura 6. Documentación y código del proyecto plugin-eskid3 y su relación. (iconos diseñados por Dooder, Archjoe y Freepik) 38

Figura 7. Fases del generador “plugin-eskid3” hecho con Yeoman. 40

Figura 8. Visualización de barras en plugin-eskid3-exampleBars 41

Figura 10 - Respuesta de ElasticSearch una vez instalado y en ejecución 52

Figura 11. Panel “Management” para insertar un nuevo índice en el panel de Kibana.
..... 53

Figura 12. Situación de botón “añadir nuevo” en el panel de administración de índices de Kibana. 53

Figura 13. Configuración de un nuevo índice en Kibana. 53

Figura 14. Selección de opción de visualización en el menú de Kibana. 54

Figura 15. Selección de una visualización en la lista ofrecida por Kibana. 54

Figura 16. Selección de un índice en el panel de Kibana para asociarlo a una visualización. 55

Figura 17. Configuración de métricas y “buckets” dentro de Kibana y visualización de barras. 55

ÍNDICE DE TABLAS

Tabla 1 - Nomenclatura Elastic vs MySQL 20

CAPITULO 1. INTRODUCCIÓN

1.1 INTRODUCCIÓN

La industria de los videojuegos ha supuesto en los últimos 40 años un gran cambio en la forma tradicional en la que pasamos nuestro tiempo libre (1). En algunos informes, se contempla que más del 70% de los niños y adolescentes de la Unión Europea y más del 90% en los Estados Unidos Juegan a videojuegos (2).

A día de hoy, los videojuegos se han convertido en una de las mayores industrias de entretenimiento, con un número de jugadores aumentando cada año gracias a la disponibilidad de nuevas consolas, plataformas y tecnologías móviles que facilitan la entrega de juegos (1)(2).

Los videojuegos generan desafíos y metas a los jugadores que provocan un aumento del interés en el uso de juegos con fines educativos (2)(3), comúnmente conocidos como “juegos serios”, y no solo porque es lo que utilizan los niños a diario, sino porque el diseño de buenos juegos está directamente relacionado con buenas experiencias educativas(2).

Por lo tanto, podemos definir los “juegos serios” como aplicaciones desarrolladas utilizando tecnologías de juegos, cuyos propósitos son distintos al puro entretenimiento (4). El término "serio", proviene de su rol de transmitir algún mensaje, ya sea conocimiento, habilidad o en general algo de contenido para el jugador (5).

Este tipo de juegos ha crecido rápidamente en la última década con un valor de 1500 millones de dólares en 2010, con una tasa de crecimiento en los últimos dos años del 100% anual, (6) debido a su gran potencial como medio para mejorar los logros en una variedad de dominios, tales como matemáticas, física, ingeniería, medicina, economía, historia y literatura(7), pero no se puede afirmar que todos los juegos serios cumplan su objetivo, ya que no todos los juegos están bien diseñados y desarrollados(6).

Para poder ofrecer un mejor diseño, hay que tener en cuenta que los jugadores no tienen idénticas experiencias de juego porque la de cada jugador es totalmente única. Por lo tanto, el análisis de la experiencia subjetiva de juego es parte crucial del proceso de diseño del juego (8).

El nivel con el que el jugador interactúa con el juego educativo, es uno de los factores claves a medir para la industria y el mundo académico, que utiliza los “juegos serios” como herramienta de aprendizaje (5).

Sin embargo, hay que tener en cuenta que los instrumentos disponibles para medir el comportamiento de los jugadores y aprender mejor de las preferencias y hábitos de cada uno de ellos, son más bien informales o limitados y carecen de una buena evaluación, produciendo resultados difíciles de interpretar(9).

Es aquí donde surge el concepto “analítica de aprendizaje” (Learning Analytics), acortado como LA, cuyo principal objetivo es recopilar las interacciones de los estudiantes con el contenido educativo, para analizar los datos obtenidos y convertir esta información en relevante y útil (7). Este proceso tiene multitud de beneficios. Por ejemplo, los profesores que utilizan esta tecnología, pueden seguir el progreso del estudiante mientras juega y así tomar decisiones de cualquier problema de aprendizaje, identificado en tiempo real. Además, al tener los datos de la actividad almacenados en un lugar centralizado, pueden comprender mejor el comportamiento y el rendimiento del estudiante en cualquier momento(7).

A su vez, la disciplina de analítica de juegos (“Game Analytics”), abarca un conjunto de técnicas ampliamente utilizadas por la industria de desarrollo de juegos para comprender mejor cómo los usuarios juegan sus juegos, encuentran errores y mejoran la experiencia de juego (10). En aras de la simplicidad, a lo largo de este TFG usaremos simplemente el término "Análisis de Aprendizaje" refiriéndonos a analítica de aprendizaje, análisis de juego (en este caso juegos educativos) o una combinación de

ambas técnicas: a nivel de visualizaciones individuales, la distinción entre LA, GA y GLA no es significativa.

1.2 MOTIVACIÓN

Es importante tener instrumentos unificados y validados para permitir a los investigadores de juegos educativos, crear buenos diseños (9). Por ejemplo, si el público objetivo de un juego educativo no se analiza correctamente antes de comenzar su diseño, el resultado suele ser un mal juego que no cumple las expectativas de su público (9). Las preferencias de juego y hábitos son un aspecto clave en este contexto, ya que determina qué tipo de juego puede ser más apropiado para cada público (9).

Hoy en día existen multitud de profesores de todos los niveles educativos y áreas de conocimiento, que utilizan actividades de aprendizaje basados en juegos serios para mejorar su forma de enseñar (10).

Una de las mayores barreras que existen a la hora de analizar datos, es la variedad de géneros y tipos de juegos que existen, lo que supone que sea difícil crear un modelo completo de LA aplicable a los juegos educativos (10).

Kibana es una herramienta que, junto a Elasticsearch, permite realizar visualizaciones de datos que ayudan a mejorar, entender y afianzar métodos de evaluación correctos, pero por defecto no viene con un gran catálogo de visualizaciones (ver figura 1).

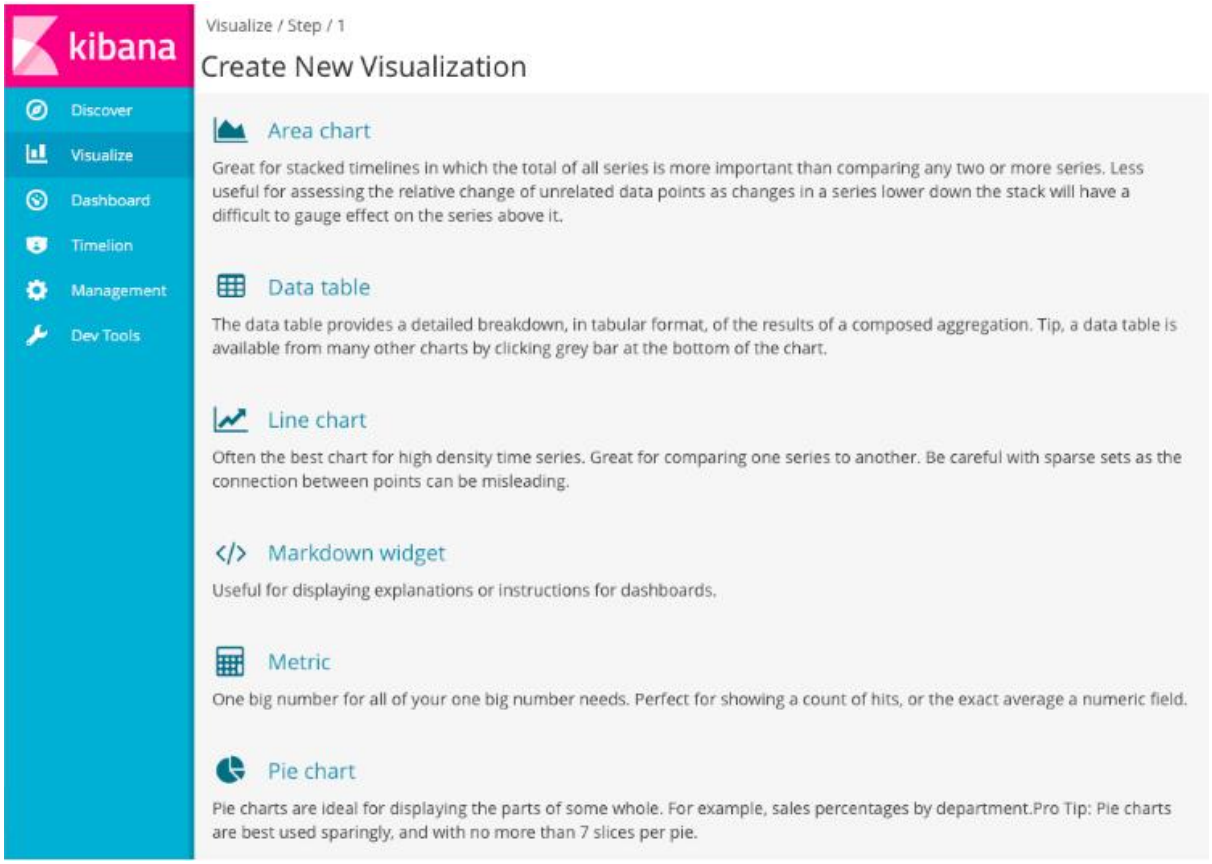

Figura 1. Visualizaciones disponibles en Kibana
(https://www.elastic.co/guide/en/kibana/current/images/tutorial-visualize.png)

El autor de este trabajo colabora con e-UCM, un grupo de 12 personas formado por personal de investigación de la Universidad Complutense de Madrid, investigadores contratados y colaboradores externos que forman parte del proyecto europeo Rage (H2020) financiado con 9 millones de euros y en el que colaboran 19 instituciones y empresas de 10 países de la Unión Europea, que busca desarrollar, transformar y enriquecer las tecnologías de la industria del juego siendo escalable, extensible y fácil de manejar para el usuario, aplicando analítica de aprendizaje en proyectos de juegos serios (11)(ver figura 2). Rage Analytics es el componente desarrollado por e-UCM en el cual, se quiere integrar plugins para Kibana, que permitan incluir nuevas visualizaciones para facilitar la analítica de aprendizaje en la plataforma. También colaboran con BEACOMING (H2020) con 16 socios y financiados con 6 millones de euros aportando una versión mejorada de Rage Analytics

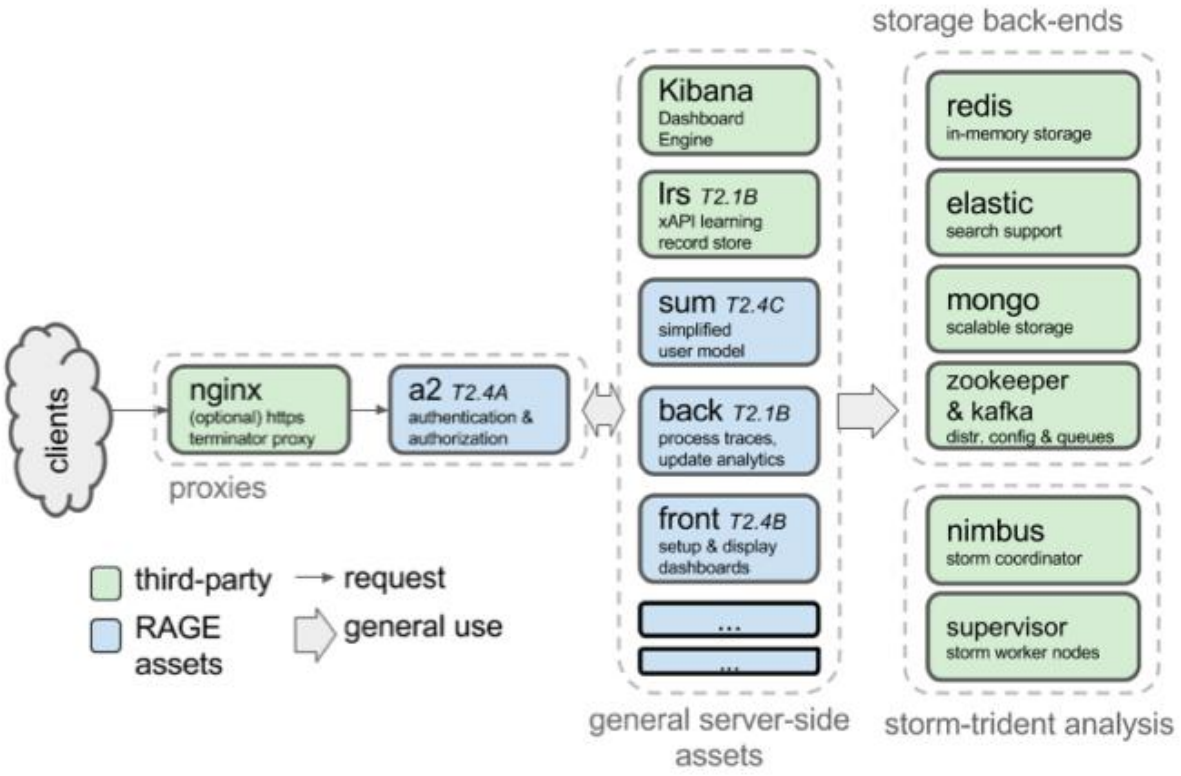

Figura 2. Infraestructura de soporte proporcionada por los componentes de analítica del proyecto RAGE (13)

1.3 OBJETIVOS

El presente Trabajo Fin de Grado (de ahora en adelante TFG), tiene como objetivo facilitar la creación de nuevas visualizaciones que sean fáciles de usar y que se integren dentro de una infraestructura de analítica aprendizaje para su uso en juegos serios.

1.4 PLAN DE TRABAJO

El proyecto se ha desarrollado en 4 fases. Cabe destacar que las tres últimas fases se realizaron de forma iterativa, en lugar de seguir un modelo en cascada. A medida que se investigó cada tecnología, se iban aplicando al proyecto y generando la documentación correspondiente en un repositorio online.

1- Definición del proyecto: Se realizaron varias reuniones con el tutor para definir y delimitar el alcance del TFG, establecer los horarios para tutorías y el correspondiente seguimiento. Se optó por utilizar un repositorio online para colaborar (Google Drive) y almacenar toda la documentación necesaria para llevar a cabo el proyecto.

2- Investigación: En esta fase se comenzó a investigar sobre el contexto del proyecto y las tecnologías necesarias para poder llevarlo a cabo. Para ello se utilizaron bases de datos de artículos de investigación, tutoriales, y documentación oficial.

3- Ejecución del proyecto: En este momento se llevó a cabo la creación de una estructura para un plugin de visualización en Kibana. También se crearon varios ejemplos y se programó un software capaz de generar una estructura básica de un plugin.

4- Documentación: Esta fase tuvo como principal objetivo generar toda la documentación del TFG.

1.5 ESTRUCTURA DEL DOCUMENTO

El resto del TFG se estructura en 4 capítulos, sin contar el capítulo 1 por ser la introducción:

- En el capítulo 2 se plantea la tecnología que se escogió para realizar el proyecto, su estructura, el diseño del plugin, la implicación técnica que tiene sobre el proyecto y la tecnología que se ha utilizado.

- En el capítulo 3 se especifica el proceso de creación de la estructura de un plugin para Kibana, el desarrollo de un programa que automatice el proceso de creación y algunos ejemplos. También describe los pasos necesarios para crear un plugin para integrar nuevas visualizaciones en un sistema de análisis de aprendizaje.

- El capítulo 4 describe los resultados obtenidos de todo el proceso de este TFG.

- En el capítulo 5 se presentan las principales conclusiones extraídas de este trabajo y las líneas de trabajo futuro.

CAPITULO 2. DISEÑO

En este capítulo, se analizan los distintos diseños posibles que permiten alcanzar el objetivo de hacer más asequibles la incorporación de nuevas visualizaciones a un sistema de LA, y se razona el porqué del diseño elegido.

2.1 VISUALIZACIONES EN LA WEB

Existen multitud de herramientas externas que permiten crear visualizaciones complejas para LA de forma muy sencilla y cómoda, consiguiendo convertir datos que en un principio son inentendibles, en información comprensible por el usuario. Además, no todas las herramientas son capaces de sincronizar de forma fácil estos datos con la visualización. La mayoría de estas bibliotecas utilizan como lenguaje de programación javascript ya que permite añadir componentes o datos en tiempo real. Algunas de estas bibliotecas son: Flot.js, Chart.js, Raphael.js, Google Charts y Highcharts.js.

En este TFG se utilizará D3.js porque tiene miles de ejemplos proporcionados por el propio autor[1], incluso el principal autor de esta biblioteca (Mike Bostock) ha creado visualizaciones para el diario estadounidense "The New York Times"[2] ganador de 108 Premios Pulitzer hasta 2012 y con más de 571,500 suscriptores (a 18 de mayo de 2017). D3 tiene una comunidad muy activa con 119 contribuidores y más de 16700 *forks* (a día 26 de abril de 2017) y documentación completa en la que se puede encontrar un manual detallado de cómo funciona. Además, D3.js se preocupa por seguir estándares web que permiten sacar todo el rendimiento de los navegadores modernos sin dejar atrás las versiones antiguas, con el objetivo de poder ejecutar las visualizaciones en cualquier navegador independientemente del sistema y versión que se estén utilizando.

[1] Más de 1000 a día 26 de abril de 2017
[2] https://github.com/d3/d3/wiki/Gallery#the-new-york-times-visualizations

Almacenando datos para su visualización

El término NoSQL fue utilizado por primera vez por Carlo Strozzi en 1998, pero no ha sido hasta 2009 cuando se empieza a popularizar por el importante crecimiento de sistemas de gestión de datos distribuidos que abandonan el soporte a las transacciones ACID (Atomicidad, consistencia, aislamiento y durabilidad) que es lo que se utiliza comúnmente en los sistemas de gestión de bases de datos relacionales (RDBMS)(12). Cómo estos sistemas se crearon y diseñaron por la necesidad de tener entornos distribuidos (almacenado en un clúster o la nube), tienen la necesidad de estar optimizados para poder manejar gran cantidad de datos, asegurando la escalabilidad horizontal y la fiabilidad de los sistemas mediante el mantenimiento de varias copias de datos. Este tipo de almacenes de datos simplifican el desarrollo de aplicaciones que requieren acceso concurrente a los datos, algo que es importante a la hora de realizar visualizaciones (12).

Existen muchos sistemas de gestión de datos NoSQL populares, como por ejemplo, MongoDB. En este proyecto no se utiliza este sistema, porque el tipo de consultas que proporciona son menos sofisticadas que las de Elasticsearch (12). Elasticsearch es un software de código abierto[3] que permite almacenar los datos recibidos sin estructura, por lo que no hace falta tener pensado qué tipo de datos se van a almacenar en el sistema gracias a la utilización de índices. Además, se basa en bases de datos de tipo distribuida porque los datos pueden estar almacenados y ser consultados en distintos sistemas, siendo totalmente transparente para el usuario. Esta herramienta permite utilizar diferentes tipos de bases de datos siempre y cuando sean Elastic y que colaboren entre sí, para mostrar resultados a demanda en cada momento con solo una petición. Una de las grandes ventajas de Elasticsearch, es que permite acceder de forma rápida y sencilla a los datos y hacer consultas sobre ellos en tiempo real, por lo que se puede acceder de forma inmediata a la información, siendo una solución

[3] software distribuido y desarrollado libremente. Se focaliza más en los beneficios prácticos (acceso al código fuente) que en cuestiones éticas o de libertad que tanto se destacan en el software libre.

excelente para aplicarla en LA mediante visualizaciones para poder ver el resultado de forma inmediata.

Dashboards

Un dashboard es una aplicación que permite gestionar visualizaciones y representar de forma cómoda, información con ayuda de un motor de búsqueda (como Elasticsearch), ayudando por tanto a interpretar fácilmente la información que le interesa al usuario en un momento dado.

La mayoría de los dashboard que existen para realizar visualizaciones llevan asociados una motor de búsqueda, con el objetivo de recoger datos de forma eficiente, rápida y sin gran esfuerzo, para que las visualizaciones se realicen de forma óptima, como Graphite con Carbon, Graphana con InfluxDB y Kibana con Elasticsearch (ver figura 3). Todas estas opciones son buenas, pero Kibana tiene un sistema de plugins implícito que permite incluir nuevas visualizaciones dentro del dashboard de forma sencilla, con muy buena documentación en comparación con las demás opciones. Además, la combinación Grafana e InfluxDB está más orientada a mostrar mediciones cuantitativas (como por ejemplo, medir la velocidad de trabajo de una CPU o la utilización de un periférico), mientras que Kibana proporciona más flexibilidad a la hora de explorar los datos, es fácil de instalar y tiene una gran comunidad en Github[4]. Por todas estas ventajas, se eligió utilizar en el sistema de analítica del proyecto H2020 RAGE.

[4] 179 contribuidores y más de 15200 aportaciones a día 26 de abril de 2017.

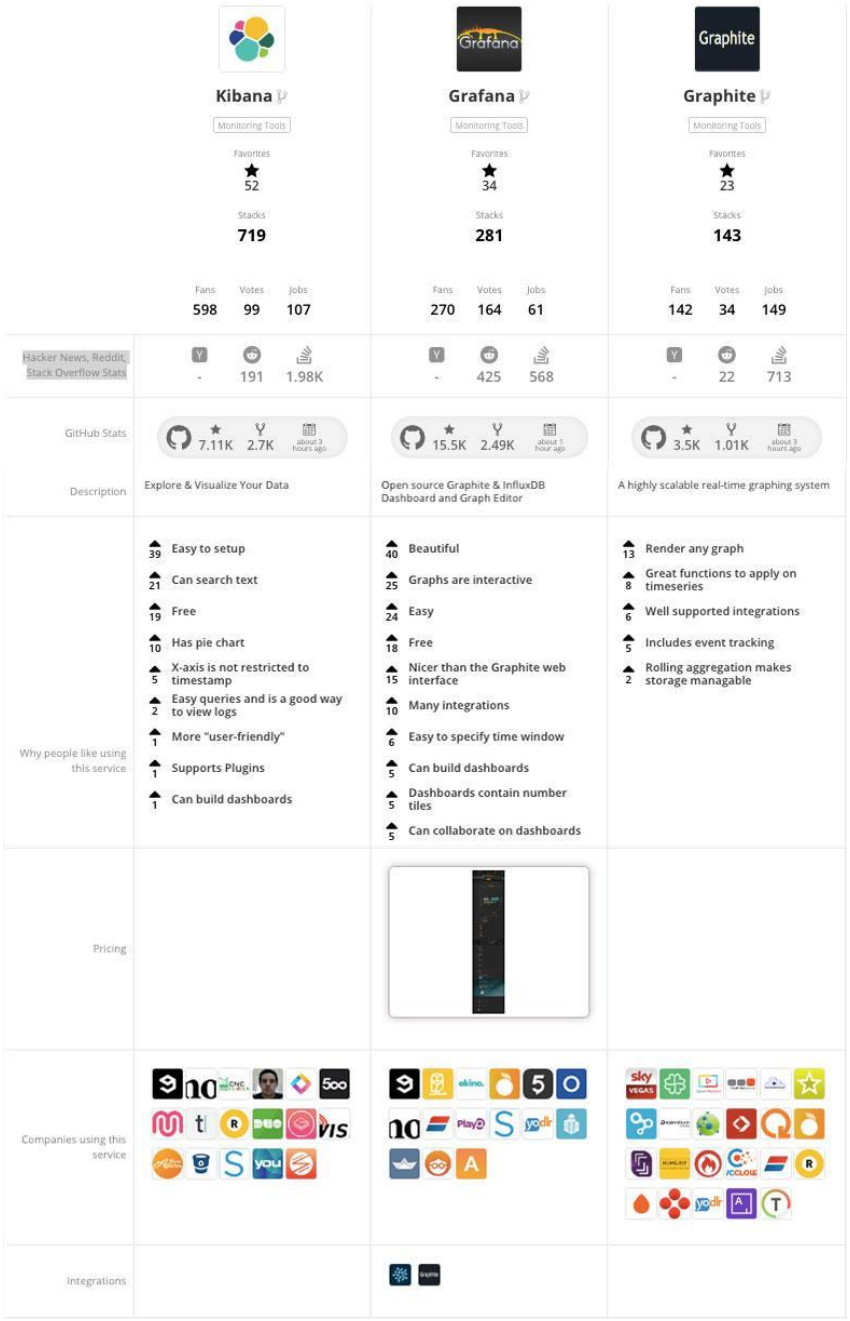

Figura 3. Comparativa Kibana - Grafana- Graphite. (https://stackshare.io/stackups/graphite-vs-grafana-vs-kibana)

NodeJS

NodeJS (de ahora en adelante Node) es un entorno multiplataforma, de código abierto[5], para la capa del servidor (pero no limitado a ello), basado en el lenguaje de programación Javascript, asíncrono[6], con entrada-salida de datos en una arquitectura orientada a eventos basado en el motor V8 de google y que promueve la utilización de módulos. Estos módulos tienen archivos donde se declaran clases, propiedades y métodos entre otros. Normalmente estos módulos son privados a otros módulos, es decir, no se puede acceder a ellos desde otros archivos directamente.

La idea principal de Node es el uso "no bloqueante", utilizar entrada-salida de datos de forma ligera y eficiente para un uso intensivo de datos en tiempo real en aplicaciones que se ejecutan en dispositivos distribuidos (que pueden estar en varios sistemas), como por ejemplo, Elasticsearch, y es una una tecnología idónea para utilizarla en LA al agilizar la obtención de datos, que se aplican a las visualizaciones web. Por todo esto, Kibana utiliza Node para implementar sus plugins y usaremos la siguiente estructura básica obligatoria necesaria para crearlos:

[5] Software distribuido y desarrollado libremente. Se focaliza más en los beneficios prácticos (acceso al código fuente) que en cuestiones éticas o de libertad que tanto se destacan en el software libre.

[6] La programación asíncrona establece la posibilidad de hacer que algunas operaciones devuelvan el control al programa llamante antes de que hayan terminado mientras siguen operando en segundo plano.

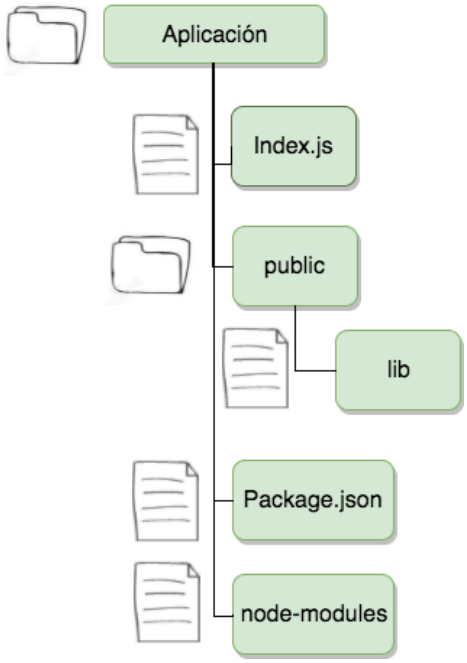

Figura 4. Estructura básica de un proyecto NodeJS.

- **Index.js**: Este archivo se utiliza para iniciar la aplicación y definir configuraciones específicas de la aplicación.
- **public:** Este directorio es la raíz de la aplicación y el sitio adecuado donde ubicar los archivos estáticos (como por ejemplo, las hojas de estilos CSS, los ficheros con código fuente escrito en lenguaje javascript donde se programa la lógica de la aplicación, los HTML donde está maquetada la estructura de la aplicación, etc).
- **lib:** Directorio donde almacenar librerías externas a la aplicación. Este directorio está dentro de public ya que también almacena ficheros estáticos.
- **package.json:** Este archivo almacena los principales parámetros de configuración del proyecto Node, tales como:
    - Nombre del proyecto.
    - Autor.
    - Versión.
    - Dependencias de la aplicación.
    - Scripts de lanzamiento.
    - Configuración del repositorio Git.

- **node_modules:** Este directorio almacena el contenido de las dependencias necesarias para que la aplicación funcione. Todas estas dependencias son las especificadas en el archivo package.json.

Node Package Manager (NPM)

Este sistema facilita a los desarrolladores compartir y reutilizar código generado como paquetes o a veces también denominados módulos. Al crear un proyecto Node con NPM, crea por defecto un archivo llamado “package.json” donde viene toda la configuración previa necesaria para que un proyecto Node pueda ejecutarse.

CommonJS

CommonJS es un estándar que se encarga de proteger la privacidad de los módulos NodeJS, evitando exponerlos a un contexto global, permitiendo el acceso solo a los módulos conectados con el mismo.

Plugins

En el contexto de la informática se entiende como una aplicación que añade nueva funcionalidad a otra ya existente. Esta nueva funcionalidad complementaria suele ser ejecutada por la aplicación principal, e interactúan por medio de la interfaz. En el caso de este TFG, podemos incluir funcionalidad en Kibana añadiendo nuevas visualizaciones, que se adapten a las necesidades del usuario y así emplearlas en LA.

Scaffolding

El uso de generadores es una técnica que ayuda a crear la estructura de un proyecto o programa de forma rápida y sencilla, con el objetivo de ahorrar tiempo e implementar

buenas prácticas de programación. Esta técnica permite que el programador especifique una serie de parámetros de la aplicación que se va a generar, como por ejemplo, el nombre de la aplicación, la versión que se quiere utilizar, una descripción o incluso seleccionar qué dependencias se quieren instalar de un listado dado, etc.

Realizar un plugin para Kibana sigue siempre un mismo esquema, por lo que se puede automatizar utilizando un generador de proyectos como Yeoman. Estos generadores pasa por distintas etapas que siguen el siguiente orden:

1. **Análisis del entorno:** En este apartado el programa analiza si el sistema donde se está utilizando cumple todos los requisitos.
2. **Recopilación de datos:** En esta fase se lanzan una serie de preguntas de configuración necesarias para generar el proyecto, como por ejemplo, el nombre del proyecto, una breve descripción, que versión utiliza, etc.
3. **Creación de estructura:** En esta tercera fase se crea la estructura del plugin utilizando las características que se han introducido en la fase anterior.
4. **Instalación de dependencias:** Esta es la última fase. En ella se instalan las dependencias necesarias que tiene el plugin de forma automática.

Una vez se ha pasado por todas las fases e instalado las dependencias, se tiene la arquitectura de un proyecto completamente construido.

2.2 DISEÑO ELEGIDO

En este apartado se detallan los pasos obtenidos para poder alcanzar los objetivos planteados en este TFG:

- Se elige Elasticsearch y Kibana para analizar y entender los datos recogidos (10). Se usa Elasticsearch como motor de búsqueda distribuido y Kibana para visualizar y explorar los datos.
- Se usa D3.js para crear visualizaciones. Para ello, hay que analizar visualizaciones ya hechas y comprender de qué manera han sido creadas para entender cómo crear visualizaciones propias.
- Se estudia la generación y configuración de proyectos Node con NPM, ya que los plugins en Kibana se basan en este sistema de gestión de paquetes para Node.
- Crear una plantilla para generar plugins de tipo visualización para Kibana y subirlo a un repositorio Github con su correspondiente documentación (wiki).
- Desarrollar un generador que sea capaz de crear una estructura de un plugin de forma automática para visualizar en Kibana y subirlo a un repositorio Github.
- Construir un repositorio de código en la plataforma Github donde almacenar todos los programas y toda la documentación creada en este trabajo fin de grado.

2.3 SOLUCIÓN ADOPTADA

Kibana y Elasticsearch

Kibana es una plataforma de análisis de datos que utiliza visualizaciones para facilitar la comprensión de los datos, y ayudar al usuario a extraer conclusiones. Es un proyecto de código abierto[7] y diseñada para trabajar en una capa superior al motor de

[7] software distribuido y desarrollado libremente. Se focaliza más en los beneficios prácticos (acceso al código fuente) que en cuestiones éticas o de libertad que tanto se destacan en el software libre.

búsqueda Elasticsearch. Fue desarrollada por Elastic, al igual que Elasticsearch, con el objetivo de tener una interfaz basada en un navegador flexible, capaz de crear visualizaciones nuevas de forma fácil y sencilla utilizando los datos almacenados en Elasticsearch, siendo ideal para poder aplicarlo a LA en los juegos educativos. Kibana es capaz de trabajar con cantidades grandes de datos obtenidos de fuentes distintas. Una de las características más importantes de Kibana, es que los gráficos pueden cambiar con consultas realizadas en tiempo real (13).

Elasticsearch está orientado a documentos, es decir, almacena documentos completos (denominados objetos en formato JSON) e indexa el contenido de cada uno de ellos para poder buscar dentro de ellos. Para comenzar a utilizar Kibana primero es necesario tener Elasticsearch funcionando y tener contenido en la base de datos.

La estructura que utiliza Elasticsearch es la siguiente (ver tabla 1): Un documento pertenece a un tipo, y los tipos se alojan dentro de un índice (más conocido como base de datos), por lo que un clúster puede contener varios índices que a su vez contienen varios tipos (tablas). Estos tipos almacenan los documentos (filas de las tablas), cada uno con múltiples campos (columnas de la tabla). Una característica muy útil de Elasticsearch es que permite definir un mapeo de campos dentro de un índice y un tipo antes de que tengamos datos dentro del mismo. Si se realiza esto, los campos de los datos que se agregan a dicho índice y tipo concreto serán asignados a sus tipos correctos para que posteriormente sean procesados correctamente (13).

Elastic	Mysql
Cluster	Esquema
Índice	Base de datos
Tipo	Tabla

Documento	Fila
Campo	Columna

Tabla 1 - Nomenclatura Elastic vs MySQL

Plugins para Kibana

Los plugins en Kibana permiten incluir nuevas visualizaciones en un entorno ES-Kibana utilizando distintas tecnologías web, como javascript para programar la lógica de la aplicación y la librería externa D3.JS encargada de realizar las visualizaciones. Además, estos plugins siguen una jerarquía específica basada en la empleada en proyectos NodeJS descrita en el apartado 2.1.

- Registro de visualización

  Para que Kibana sea capaz de interpretar el plugin y saber que configuración aplicar, es necesario crear dos archivos importantes basados en módulos NPM:

  A. **Package.json:** Este archivo se encuentra en la raíz del proyecto y en él se detallan las características del plugin, como el nombre, la versión y las dependencias necesarias para que el plugin funcione correctamente.
  B. **Index.js:** Este segundo archivo contiene el módulo principal del plugin encargado de iniciarla y la descripción del tipo de plugin que se desea crear, en este caso, tipo visualización.

  Además de estos dos archivos que forman la estructura básica de un plugin en Kibana, también es importante tener una carpeta con nombre **"public"** en el mismo directorio que los ficheros anteriores, en la que se almacenan los archivos en los que se programara toda la lógica de la aplicación (los fichero estáticos).

- Definir la visualización

  Para que Kibana sepa las características del plugin, como el icono que aparecerá en la lista de visualizaciones disponibles en Kibana, el nombre, el título y una pequeña descripción, es necesario incluir un archivo javascript dentro de la carpeta *public* (ya que se trata de un fichero estático) donde se especifican todos estos parámetros, como la plantilla o vista en formato HTML donde se va a colocar la visualización (este HTML se deberá colocar en la carpeta public al pertenecer a la lógica de la aplicación y ser estático). Este archivo tiene que tener el mismo nombre que el colocado en el archivo "package.json".

- Programar un controlador

  Para poder utilizar los datos demandados a ES desde Kibana, es necesario incluir dentro de la carpeta *public,* un archivo javascript donde se programe el controlador encargado de recibir los datos de Elasticsearch y filtrar el contenido para coger solo los datos que se vayan a utilizar. Todo este proceso está más detallado en el Anexo A.

- Visualizar datos.

  Una vez se tienen todos los datos filtrados en el controlador, se necesita incluir una librería de visualizaciones que permita transformar estos datos en información entendible por el usuario. En este caso, D3 permite incluir visualizaciones que utilizan el DOM del HTML que se está utilizando para generar visualizaciones complejas de forma sencilla. Este archivo javascript incluye la librería D3.js, y se debe almacenar en el directorio *lib* de la carpeta *public*. Este archivo permite que se pueda crear un objeto que sea capaz de utilizar D3 para generar visualizaciones, utilizando los datos filtrados previamente por el controlador, aunque es posible que también tenga que transformar los datos a como los demanda la propia visualización.

- Automatizar procesos con Yeoman

  Para automatizar la creación del esqueleto de un plugin para Kibana, se utiliza una herramienta generadora de código denominadas “scaffolding”. Estas herramientas permiten iniciar un proyecto o programa de forma rápida, sencilla y preservando buenas prácticas, con el principal objetivo de evitar tareas repetitivas. Para crear un generador que automatice la creación de plugins para Kibana, utilizaremos Yeoman, una herramienta de código abierto creada bajo una gran comunidad (más de 700 generadores mantenidos por la comunidad[8]) y construida con tecnología ,

  El flujo de trabajo que realiza Yeoman es la combinación de 3 herramientas que en conjunto logran su cometido:

  - Bower: para el manejo de dependencias
  - Grunt: para la previsualización, ejecución de pruebas y construcción de la aplicación.

  Y por último:

  - **Yo:** para construir el esqueleto de la aplicación dependiendo de su tipo.

  Existen miles de generadores[9] ya creados en el repositorio de Yeoman[10] que se pueden instalar y utilizar. Un generador es una rutina escrita para construir un esqueleto en particular. Es importante realizar la instalación de cualquier generador con la opción -g en el caso de que no se quiera instalar en un proyecto en concreto.

  En este TFG se desarrolla un generador con el fin de agilizar la creación de plugins para Kibana.

[8] http://yeoman.io/community-generators.html
[9] 6150 a día 14 de mayo de 2017
[10] http://yeoman.io/generators/

CAPITULO 3. DESARROLLO

Ya que el objetivo principal de este TFG, es simplificar al máximo la creación de nuevas visualizaciones para LA, a continuación se detallan los pasos necesarios para, partiendo de una instalación limpia, añadir un plugin de visualización en Kibana.

3.1 PROCESO DE CREACIÓN DE UN PLUGIN

Para explicar este proceso se utiliza la estructura básica de un plugin para Kibana "plugin-eskid3" creada y almacenada en el github del autor de este TFG (ver figura 5).

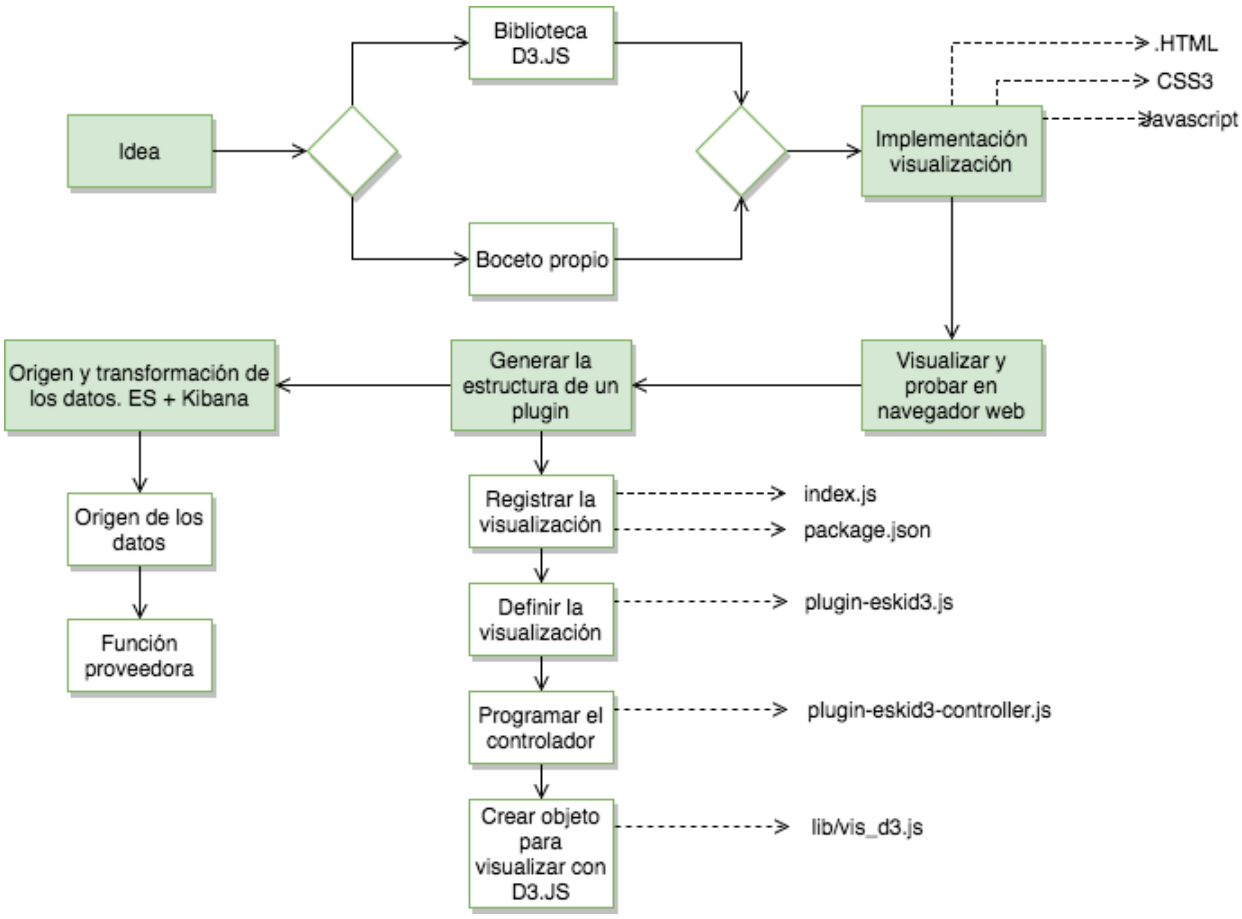

Figura 5. Proceso de creación de un plugin para Kibana.

3.1.1 Diseño de la visualización

- **Boceto**

  Lo primero que se ha de hacer, es pensar qué tipo de visualización se adapta mejor a lo que se quiere representar. Una vez se tiene claro qué tipo se va a utilizar, es aconsejable hacer un boceto de cómo se quiere que sea la visualización. En el caso de que no se quiera crear la visualización desde cero, se puede consultar la biblioteca de visualizaciones de D3.js disponible en la documentación, y escoger alguna que se adapte a las necesidades.

- **Implementación**

  Para poder desarrollar la visualización, es aconsejable utilizar herramientas online como "jsfiddle" que permite crear código HTML, CSS3 Y Javascript visualizando el resultado en un navegador web. Todo este proceso sigue un desarrollo rápido de aplicación (RAD), en el que se va adaptando el diseño de la visualización en periodos de tiempo cortos, para ahorrar tiempos de desarrollo. Por ello, una vez se tiene funcionando la visualización y probada, se puede comenzar a crear la estructura del plugin donde se utilizará esta visualización.

3.1.2 Requisitos mínimos y creación de un plugin básico

Para poder crear plugins en Kibana, se deben cumplir una serie de requisitos previos y tener descargado y configurado correctamente Elasticsearch y Kibana (Ver Anexo B).

Para facilitar la creación de la estructura del plugin, se puede utilizar el generador Yeoman que se puede descargar[11] en el repositorio github del autor de este TFG ("generator-plugin-eskid3"), donde viene descrito como se puede utilizar para generar la estructura del plugin.

[11] https://github.com/tomas-teston/generator-plugin-eskid3

A continuación, se detallan los pasos a seguir para crear un ejemplo de la estructura de un plugin para Kibana, que es capaz de recibir datos de Elasticsearch para mandarselos a cualquier visualización que se quiera utilizar:

A. Registro de la visualización

Los plugins en Kibana siguen una estructura de proyecto tipo NodeJS por lo que se debe configurar los siguientes dos archivos: package.json e index.js:

a. **Index.js**: Para que Kibana sepa el tipo de plugin que se está implementando, se coloca el tipo de plugin que se va a utilizar, en este caso es de tipo visualización.

b. **Package.json:** Este archivo es un módulo NPM necesario para que el proyecto NodeJS funciona correctamente. En este archivo se incluyen varios aspectos de configuración importantes, como el nombre del plugin, la versión, y las dependencias necesarias para su correcto funcionamiento. En este caso el nombre será “plugin-eskid3” y la versión 6.0.0.

B. Definir la visualización

Una vez se haya hecho la configuración previa del proyecto, el siguiente paso es configurar el propio plugin. Para ello, se crea un archivo javascript con el mismo nombre que se haya colocado en el archivo “package.json” dentro de la carpeta public, donde se coloca el nombre, título, descripción, etc. Esta configuración aparecerá en el listado de visualizaciones disponibles dentro del listado de Kibana.

C. Programar el controlador

El controlador “plugin-eskid3-Controller”, almacenado en la carpeta Public, es el encargado de recibir los datos de Elasticsearch y mandarselos a la visualización. Este controlador utiliza módulos Angular que facilitan la sincronización entre Elasticsearch y el plugin. Además, también se sincroniza la

vista que se utilizará para mostrar la visualización con el método Angular *controller*.

Lo primero que se programa en este controlador, es el filtro encargado de localizar el contenido de los datos que interesen dentro de la respuesta que devuelve ES.

Para poder recibir los datos de ES se utiliza el método $watch, proporcionado por $scope de Angular el cual, espera a que ES mande los datos en la variable (*esResponse*) y, utilizando el filtro anterior, se filtran los datos para recoger solo los que interesen.

Por último, se incorpora el módulo de la librería D3.js que se encarga de realizar las visualizaciones con los datos que se le transfieren desde el controlador.

Todo este proceso se describe más detallado en el Anexo A.

D. Objeto para visualizar con D3.

El objeto javascript creado en la carpeta "lib" en "public", es el encargado de adaptar los datos para que sean interpretables por la visualización que se incorpore dentro de la función "render". En este ejemplo la función render aparece vacía ya que es un ejemplo sin visualización.

3.1.3 Origen y transformación de los datos. Elasticsearch y función proveedora

- **Origen de los datos**

  En el caso de un ejemplo real, todos los datos que se encuentran en Elasticsearch son proporcionados por los juegos serios que han sido programados para almacenar cierta información acerca de cómo interactúan los jugadores con el

juego. Por ejemplo, un juego para primeros auxilios en el que se están realizando tareas de reanimación, puede estar mandando mediciones de cómo está actuando el alumno frente al paciente. De esta manera, se puede aplicar LA realizando visualizaciones para comprender la información que se ha almacenado.

- **Función proveedora**

  Dentro del controlador del plugin existe una función AngularJS encarga de recibir y transformar los datos que se solicitan desde el. Esta función AngularJS es *$watch,* que recibe como parámetro el resultado de la consulta y, una vez se recibe los datos de la consulta (también denominados "buckets"), se filtran para dejar solo aquellos que interesen.

3.2 EJEMPLO BÁSICO APLICADO DE UN PLUGIN

Este ejemplo utiliza como fuente de datos la ofrecida por la documentación de Kibana (Shakespeare.json), que almacena multitud de datos acerca de las obras de Shakespeare. Se pueden hacer consultas sobre esta base de datos para obtener información, como por ejemplo, que obras de shakespeare son las que más líneas tiene. Para ello, se utiliza un diagrama de barras simple recogiendo los datos de Elasticsearch. Para realizar este plugin hay que tener en cuenta que sigue la estructura del proyecto base "plugin-eskid3". Este plugin está en el github del autor de este TFG con nombre "plugin-eskid3-Bars"[12].

3.2.1 Diseño de la visualización

Para poder llevar a cabo este plugin, se realiza una visualización de barras en la herramienta online jsfiddle con ayuda de la librería D3.JS.

[12] https://github.com/tomas-teston/plugin-eskid3-exampleBars

3.2.2 Cargar datos en Elasticsearch

Los datos de ejemplo necesarios para hacer esta prueba, son los proporcionados por la documentación oficial[13] de Kibana (shakespeare.json).
Una vez se tienen los datos se realizan las siguientes acciones, con ayuda de una herramienta de peticiones a servicios REST:

1. Lanzamos Elasticsearch.
2. Se crea un "index" nuevo en Elasticsearch: Para ello, se ejecuta una petición de tipo "POST" indicando el nombre al final de la petición: http://localhost:9200/shakespeare.
3. Por último, añadimos los datos descargados anteriormente lanzando una petición "POST" colocando la siguiente configuración:
    a. Cabecera: content-type: application/json.
    b. Contenido: shakespeare.json

3.2.3 Registro de la visualización

Para comenzar se crean los dos archivos principales: package.json e index.js:

A. **Index.js**: En este archivo se coloca el tipo de plugin que se va a utilizar, en este caso es de visualización.
B. **Package.json:** En este archivo se definen las dependencias necesarias, los comandos para poder enlazar el plugin con Kibana y distintos elementos de configuración como el nombre del plugin que en este caso se define como "plugin-eskid3-bars".

[13] https://www.elastic.co/guide/en/kibana/current/tutorial-load-dataset.html

3.2.4 Definir la visualización

Una vez se haya hecho la configuración previa de la visualización, el siguiente paso es definir un archivo de javascript dentro de la carpeta public (en este caso se llama "plugin-eskid3-bars.js"), donde se realiza la configuración de la información que Kibana mostrará en el listado de visualizaciones, como el nombre, título y la descripción.

3.2.5 Programar el controlador

El controlador es el encargado de recibir los datos de Elasticsearch y mandarselos a la visualización. Este está programado en el archivo almacenado en la carpeta Public "plugin-eskid3-bars-Controller". Para poder llevarlo a cabo, este archivo dispone de un método Angular llamado $watch que es llamado como respuesta cuando se realiza una consulta a Elasticsearch. Este método recibe como parámetro los datos ("esResponse"), que son los enviados al objeto encargado de realizar la visualización.

3.2.6 Objeto para visualizar con D3

Este objeto es el encargado de parsear los datos, tal y como los pide la visualización elegida de la librería D3.JS que se utiliza y enlazarlos. En este ejemplo, la visualización utilizada ha sido construida por el propio autor de este TFG.

Para conocer el proceso de prueba de un plugin en Kibana ver el anexo C.

3.3 EJEMPLO AVANZADO DE UN PLUGIN

En este ejemplo se muestra una visualización con jerarquía de árbol. Para el ejemplo se utilizan datos ficticios creados por el autor de este TFG. Estos datos recrean la relación que existe entre los jugadores de un juego y los niveles que han superado. Gracias a esto, el creador del juego puede saber en qué nivel dejan de jugar la mayoría de sus usuarios y poder sacar conclusiones. Este plugin sigue la misma estructura que el proyecto base “plugin-eskid3”.

3.3.1 Diseño de la visualización

Este plugin se basa en una visualización de tipo árbol ofrecida por la biblioteca oficial de la librería D3.JS. El primer paso es llevar esta visualización a jsfiddle donde se comprueba el funcionamiento de la misma, teniendo en cuenta la jerarquía de datos que solicita la visualización necesaria para poder mostrar los datos.

3.3.2 Cargar datos en Elasticsearch

Los datos de ejemplo necesarios para hacer esta prueba se introducen en Elasticsearch siguiendo las siguientes introducciones:

1. Lanzamos Elasticsearch.
2. Se crea un “index” nuevo en Elasticsearch: Para ello, se ejecuta una petición de tipo “POST” indicando el nombre al final de la petición: http://localhost:9200/game_level.
3. Por último, añadimos los datos lanzando una petición “POST” colocando la siguiente configuración:
    a. Cabecera: content-type: application/json.
    b. Contenido (Realizar tantas peticiones como trazas queremos tener en la base de datos):

```
{
        "username": "nombre_jugador",
        "level": 1
}
```

3.3.3 Registro de la visualización

Para comenzar se crean los dos archivos principales: package.json e index.js:

A. **Index.js**: En este archivo se coloca el tipo de visualización que se va a utilizar, en este caso es de tipo visualización.
B. **Package.json:** En este archivo se incluyen varios aspectos de configuración importantes, como el nombre del plugin, la versión, y las dependencias necesarias para su correcto funcionamiento. En este ejemplo el parámetro nombre se define como “plugin-eskid3-tree”.

3.3.4 Definir la visualización

Una vez se haya hecho la configuración previa de la visualización, el siguiente paso es definir un archivo de javascript dentro de la carpeta public (en este caso se llama“plugin-eskid3-tree.js”), donde se realiza la configuración de la visualización, como el nombre, título, descripción, etc. Esta configuración aparecerá en el listado de visualizaciones disponibles dentro del listado de Kibana.

3.3.5 Programar el controlador

Este controlador dispone de un método Angular llamado $watch, que es llamado como respuesta cuando se realiza una consulta a Elasticsearch. Este método recibe como parámetro los datos (“esResponse”), que son los enviados al objeto encargado de realizar la visualización. En este método existen diferencias con respecto al del plugin “plugin-eskid3”, ya que en este caso se reciben dos buckets en vez de uno. Para poder

ajustar el controlador, se pasa como valor del elemento “value” un conjunto de elementos asociados.

3.3.6 Objeto para visualizar con D3

Este objeto es el encargado de adaptar los datos recibidos a la forma solicitada por la visualización en cuestión y enlazarlos.

CAPITULO 4. RESULTADOS

Durante el desarrollo de este TFG, se han creado visualizaciones para Kibana agilizando el proceso con la ayuda de un generador de código, y una biblioteca de visualizaciones para facilitar y perfeccionar la inclusión de estas en dashboards, utilizados para LA en juegos serios.

Este generador de plugins para Kibana, el plugin base y los demás ejemplos construidos durante el TFG, se encuentran almacenados en el repositorio Github del autor.

Este repositorio se estructura de la siguiente manera:

4.1 PLUGIN-ESKID3

Este proyecto almacena el código y la documentación necesaria para crear una estructura principal de un plugin en Kibana (ver figura 6).

Este proyecto contiene:

- Una guía rápida inicial donde se describe cómo podemos configurar el entorno necesario para empezar utilizar un plugin y cómo utilizarlo.
- El código del propio plugin (300 líneas de código).
- Una documentación extendida (wiki con más de 15 apartados) donde se describe con más detalle todo el proyecto:
    - **Introducción**: En este apartado se explica al detalle que es Kibana y Elasticsearch, y como se puede utilizar para aplicar estas herramientas en LA para juegos serios.
    - **Configuración del entorno:** Para poder ejecutar plugins y ver el resultado es necesario cumplir una serie de requisitos y configuraciones que se detallan en este apartado. Como por ejemplo, como descargar, configurar Kibana y ES, cargar datos en ES, crear y seleccionar un index de forma apropiada en ES, etc.

- **Estructura básica de un plugin:** En esta sección de la documentación se realiza un desglose del plugin explicando paso a paso que realiza cada parte del mismo. por ejemplo, como registrar una visualización dentro de Kibana, cómo programar el controlador, cómo incluir visualizaciones D3 dentro del proyecto y cómo ejecutar el plugin dentro de Kibana.
- **Más información**: En este apartado se ha incluido una breve explicación de cómo configurar una variable de entorno en distintos sistemas operativos, ya que es necesario hacerlo en el proceso de configuración del entorno.

Este proyecto almacena el código y la documentación necesaria para crear una estructura principal de un plugin en Kibana (ver figura 6).

Este proyecto contiene:

- Una guía rápida inicial donde se describe cómo podemos configurar el entorno necesario para empezar utilizar un plugin y cómo utilizarlo.
- El código del propio plugin (300 líneas de código).
- Una documentación extendida (wiki con más de 15 apartados) donde se describe con más detalle todo el proyecto:
  - **Introducción**: En este apartado se explica al detalle que es Kibana y Elasticsearch, y como se puede utilizar para aplicar estas herramientas en LA para juegos serios.
  - **Configuración del entorno:** Para poder ejecutar plugins y ver el resultado, es necesario cumplir una serie de requisitos y configuraciones que se detallan en este apartado. Como por ejemplo, como descargar, configurar Kibana y ES, cargar datos en ES, crear y seleccionar un index de forma apropiada en ES, etc.
  - **Estructura básica de un plugin:** En esta sección de la documentación se realiza un desglose del plugin explicando paso a paso que realiza cada parte del mismo. por ejemplo, como registrar una visualización dentro

de Kibana, cómo programar el controlador, cómo incluir visualizaciones D3 dentro del proyecto y cómo ejecutar el plugin dentro de Kibana.

- **Más información**: En este apartado se ha incluido una breve explicación de cómo configurar una variable de entorno en distintos sistemas operativos, ya que es necesario hacerlo en el proceso de configuración del entorno.

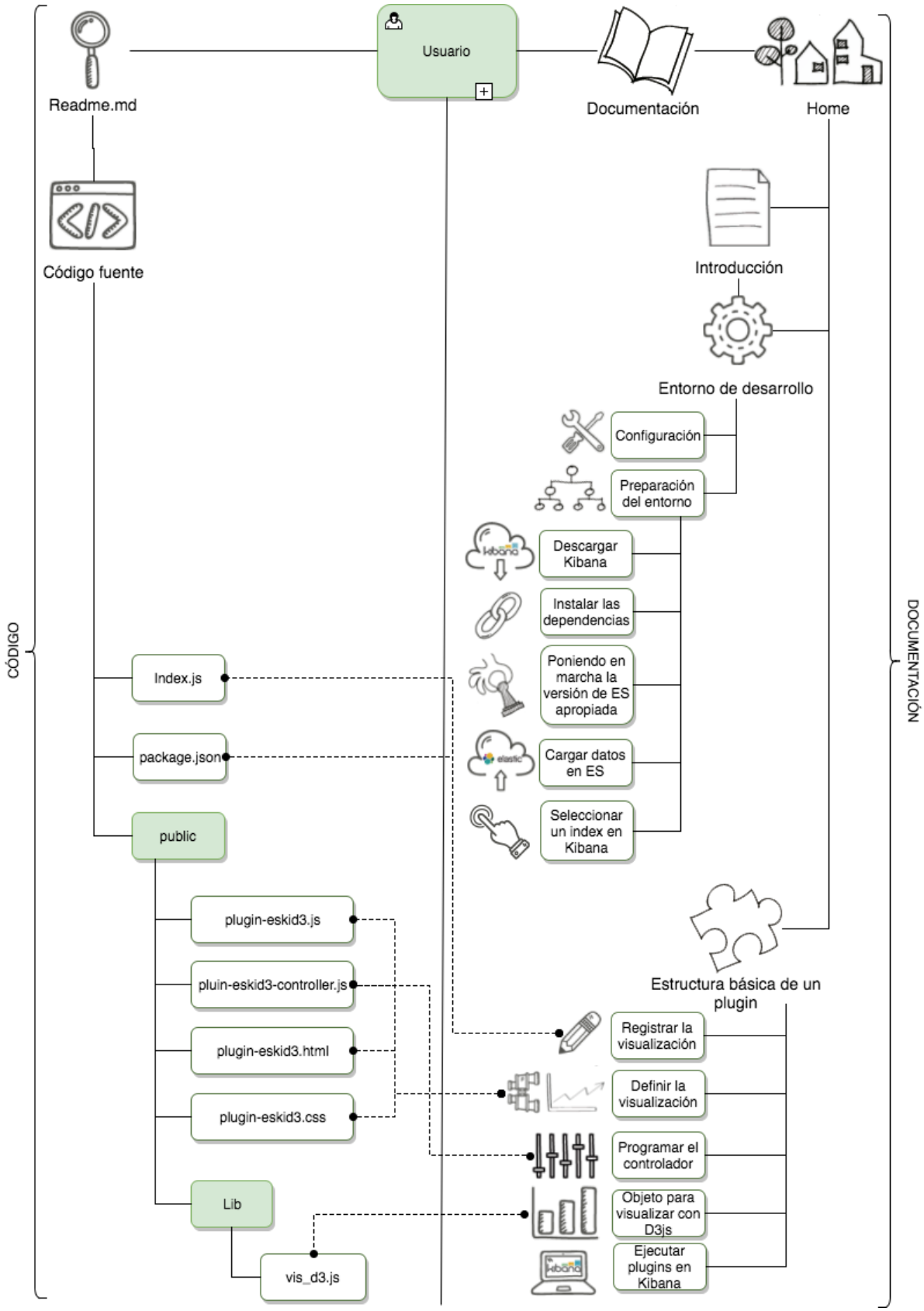

Figura 6. Documentación y código del proyecto plugin-eskid3 y su relación. (iconos diseñados por Dooder, Archjoe y Freepik)

4.2 GENERATOR-PLUGIN-ESKID3

En este proyecto almacenado en el Github del autor, se puede encontrar todo el código del generador de plugins para Kibana (500 lineas de código Javascript) y el código de las plantillas necesarias para crearlos. En el se puede encontrar una guía rápida donde se describe que es un generador Yeoman, para que se utiliza y cómo utilizarlo con unos sencillos pasos.

Este generador pasa por distintas etapas (ver figura 7) que están programadas en el archivo index.js en el siguiente orden:

1. **Analizar el entorno:** En este apartado el programa analiza si el sistema donde se está lanzando cumple todos los requisitos. El archivo encargado de analizar todo este proceso es “env.js”. En el caso de la creación de un plugin para Kibana se comprueba que exista en el mismo nivel del directorio que el proyect. También se comprueba que la versión de Node instalada en el sistema es la correcta, que java está instalado y que la versión de Kibana es compatible con la versión de Node del sistema.

2. **Fase “prompting” o fase de asistencia:** En esta fase se lanzan una serie de preguntas de configuración necesarias para generar el proyecto, como por ejemplo, el nombre del proyecto y una breve descripción. Todo esto está detallado en el archivo “prompts/index.js” el cual utiliza las preguntas programadas en el archivo “prompts/prompts.json” en formato JSON.

3. **Creación de estructura:** En esta tercera fase se detalla que archivos van a crear la estructura del proyecto, inyectando los datos necesarios que se han introducido en la fase anterior por el usuario. Estos archivos se almacenan dentro de la carpeta “templates”.

4. **Instalación de dependencias:** Esta es la última fase. En ella, se instalan las dependencias automáticamente detalladas en el archivo “package.json”, en el apartado de dependencias.

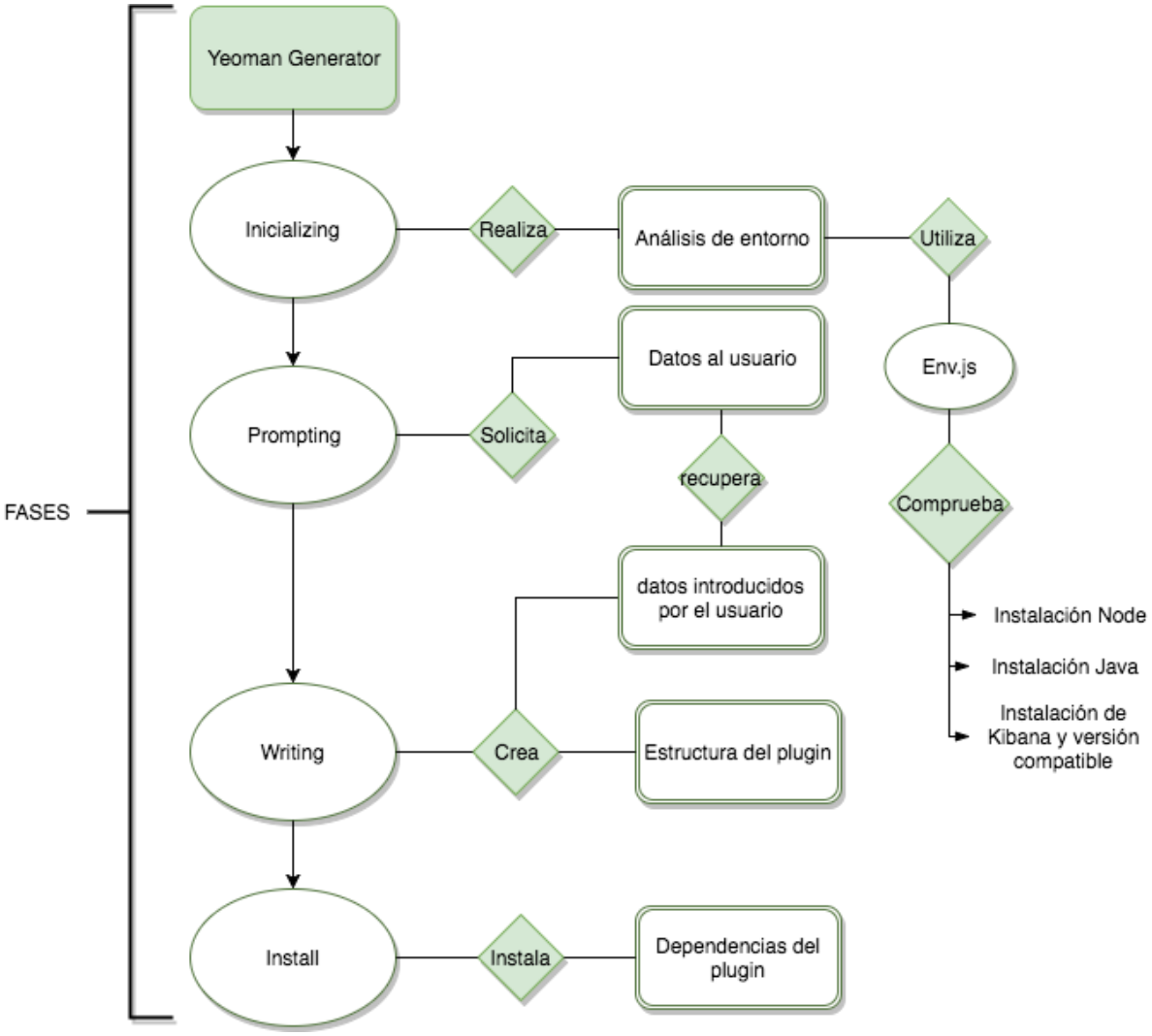

Figura 7. Fases del generador “plugin-eskid3” hecho con Yeoman.

Para instalar este generador se utiliza NPM y el siguiente comando: “npm install -g generator-plugin-kibana-eskid3”.

Dentro del directorio se ejecuta el comando “yo plugin-kibana-eskid3” y se lanza el asistente del generador del plugin, se rellenan los datos y automáticamente genera el plugin básico en el sistema.

4.3 PLUGIN-ESKID3-EXAMPLEBARS

Este proyecto contiene el código necesario (mas de 300 lineas de código javascript, HTML y CCS3), para crear un ejemplo de un plugin de visualización para Kibana, utilizando una visualización de diagrama de barras, generado por el autor del TFG con la librería D3.js. Para poder realizar las pruebas, se utiliza como fuente de datos los proporcionados por la guía de Elasticsearch (shakespeare.json).

El proyecto almacena una guía rápida en la que se puede encontrar una breve introducción de las tecnologías utilizadas en el proyecto, cómo empezar (la configuración necesaria para poder utilizarlo), y los comandos necesarios para incluir este plugin dentro de una infraestructura ES - Kibana y poder probarlo (el resultado es el mostrado en la figura 8).

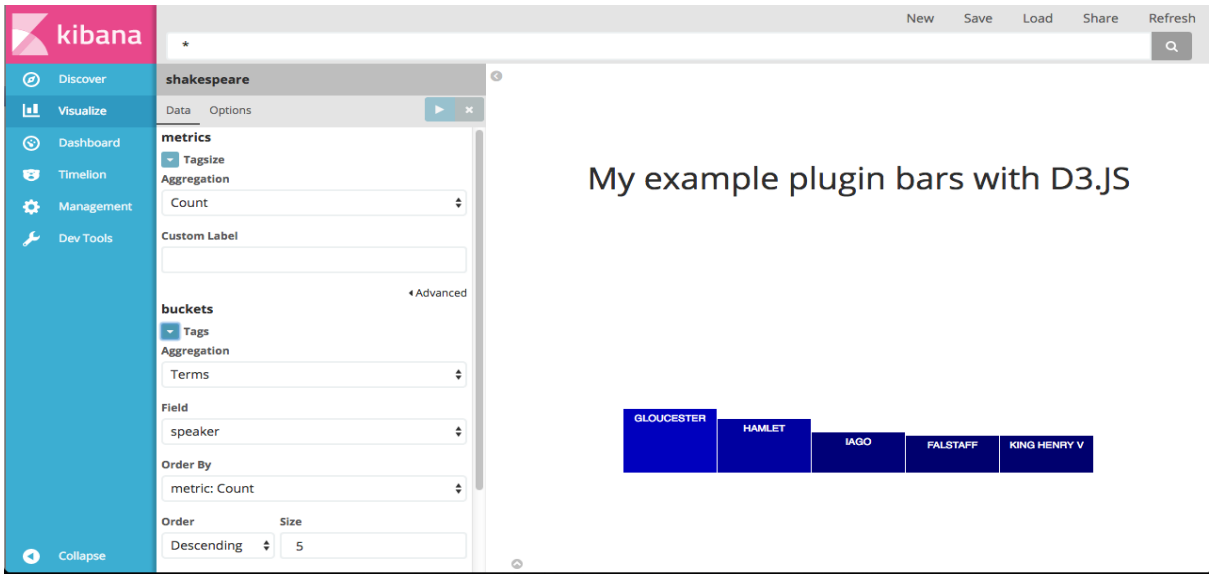

Figura 8. Visualización de barras en plugin-eskid3-exampleBars

4.4 PLUGIN-ESKID3-TREE

En este ejemplo de plugin para Kibana se describe un caso de uso real, donde se utiliza una visualización de tipo árbol seleccionada de la biblioteca que proporciona D3.js (figura 9).

Dentro de este proyecto en Github, se encuentra una guía rápida de uso donde se describe que configuración es necesaria para poder utilizar este plugin, qué estructura deben de seguir los datos para poder ser visualizados con el plugin de forma correcta, cómo empezar a utilizar el plugin y los comandos necesarios para probar la visualización dentro de un entorno ES + Kibana. Además, está disponible todo el código (más de 700 líneas) para poder descargarlo.

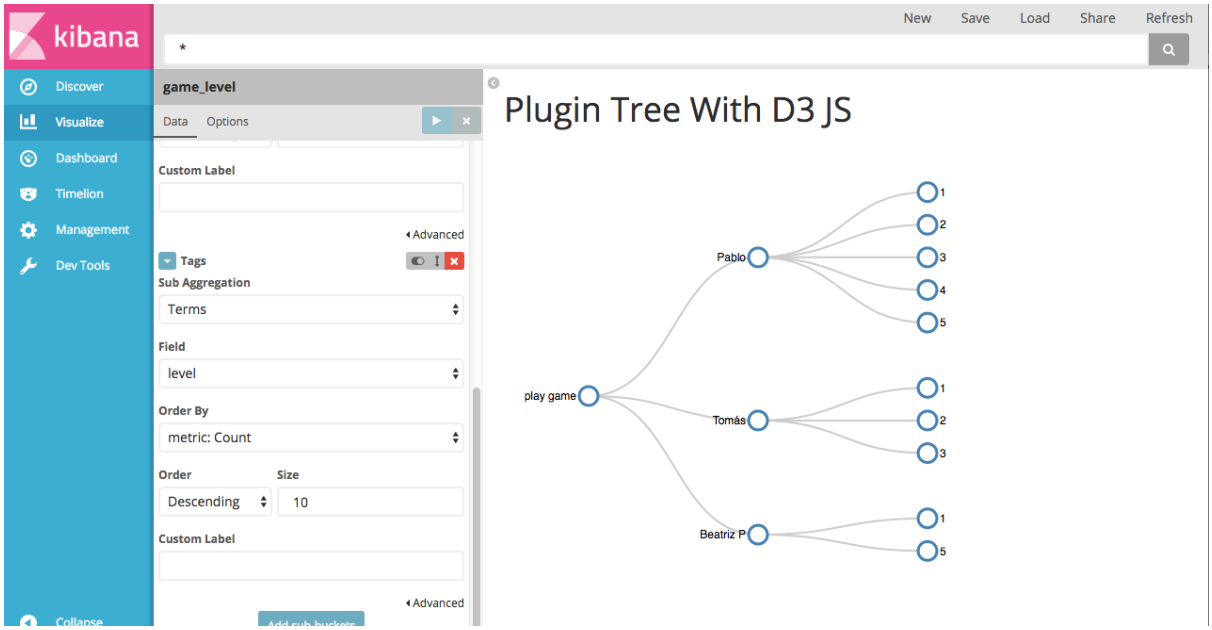

Figura 9. Visualización de tipo árbol en plugin-eskid3-tree

CAPITULO 5. CONCLUSIONES Y TRABAJO FUTURO

Para finalizar, se analizan las conclusiones y trabajos futuros que no fueron posibles incluir en este TFG.

5.1 CONCLUSIONES

Para desarrollar y entender el análisis de los juegos serios, es útil usar buenas visualizaciones para proporcionar información que sea entendible por el usuario de forma instantánea. Si aplicamos esto a juegos serios, es útil saber qué está sucediendo dentro del juego en cada momento y así, poder obtener información de cada interacción que ocurre entre el jugador y el juego.

Con todo el trabajo desarrollado en este TFG, es posible desplegar nuevas visualizaciones en Kibana que ayuden a entender mejor estos datos. Con ayuda de la librería D3.js, se consigue crear nuevas visualizaciones complejas de forma sencilla ya que proporciona una extensa biblioteca de visualizaciones ya hechas, desde las que podemos partir para crear nuestras propias visualizaciones.

Para poder realizar estos plugins, se tuvo que aprender bien cómo se estructuran los plugins dentro de la arquitectura de Kibana, tecnologías punteras AngularJs, NodeJS, Javascript, RequireJS y Github

Una vez se desarrolló el esqueleto principal de un plugin, se llegó a la conclusión de que utilizar un generador de proyectos (en este caso, plugins para Kibana), sería interesante ya que se ahorraría horas en hacer un proceso que siempre se hace de la misma manera. Por tanto, todos los plugins de ejemplo desarrollados en este TFG, se han creado utilizando un generador que se desarrolló previamente a empezar a crear ejemplos.

En este TFG se creó un plugin de prueba para incluirlo en el proyecto RAGE (en un entorno real) utilizando una visualización de la biblioteca D3.js. Todo ello se desarrolló para la versión de Kibana que se estaba utilizando en el proyecto RAGE. Se comprobó

que funcionaba correctamente dentro de RAGE, pero la visualización que se escogió no tenía licencia que permitiera redistribuirla a terceros, y por tanto no pudo ser incorporada al repositorio de código que acompaña a este trabajo

5.2 TRABAJO FUTURO

A lo largo de todo este trabajo me hubiera gustado haber podido investigar y desarrollar los siguientes campos:

- Incluir nuevos plugins en un entorno real, como por ejemplo, el proyecto Europeo RAGE Analytics, en el que se incluyan visualizaciones nuevas para aplicarlas en LA.
- Externalizar el código de las visualizaciones para que se utilicen como una dependencia Node más. De esta forma, hacemos que el plugin sea más ligero, ya que solo incorpora el código cuando se necesita.

CAPITULO 6. CONCLUSIONS AND FUTURE WORK

To conclude, we analyze the conclusions and future work that weren´t possible to be included in this TFG.

6.1 CONCLUSIONS

To develop and understand the analysis of serious games, it is useful to use good visualizations to provide information that is understandable by the user instantaneously. If we apply this to serious games, it is useful to know what is happening inside the game at each moment and thus, to be able to obtain information of each interaction that happens between the player and the game.

With all the work developed in this TFG, it is possible to deploy new visualizations in Kibana that help to better understanding this data. With the help of the D3.js library, it is possible to create new complex visualizations in a simple way since it provides an extensive library of visualizations already made, from which we can start to create our own visualizations.

In order to make these plugins, it was necessary to learn how to structure the plugins within the architecture of Kibana, leading technologies AngularJs, NodeJS, Javascript, RequireJS and Github

Once the main skeleton of a plugin was developed, it was concluded that using a project generator (in this case, plugins for Kibana), would be interesting since it would save hours in doing a process that is always made of the same way. Therefore, all sample plugins developed in this TFG have been created using a generator that was developed before beginning to create examples.

In this TFG a test plugin was created to include it in the RAGE project (in a real environment) using a visualization of the D3.js library. All of this was developed for the

version of Kibana that was being used in the RAGE project. It was verified that it worked correctly inside RAGE, but the visualization that was chosen was not licensed that it wasn´t allow to be redistribute it, and therefore could not be incorporated into the repository of code that accompanies this work

6.2 FUTURE WORK

I wish I had had the possibility of investigating and have developed the following fields:

- Include new plugins in a real environment, such as the European RAGE Analytics project, which includes new visualizations to be applied in LA.

- Externalize the display code to be used as an additional Node dependency. In this way, we make the plugin lighter, since it only incorporates the code when it is needed.

ANEXO

ANEXO A: PROCESO DETALLADO PARA CREAR UN CONTROLADOR PARA ESKID3

Todos los plugins en Kibana necesitan un controlador con el que poder obtener los datos de Elasticsearch y transformarlos para que sean entendibles por las visualizaciones D3.JS. Los pasos para programar el controlador son los siguientes:

1. El primer paso que se realiza en el controlador es crear un módulo Angular para el plugin.
2. Después, se incorpora el módulo de la librería D3.js que se encarga de realizar las visualizaciones.
3. Utilizando el módulo creado en el primer paso, se llama al método controller para instanciar un nuevo controlador, que recibe como parámetros el nombre del controlador que vamos a colocar en nuestro HTML con la etiqueta Angular NG-CONTROLLER, y una función anónima que recibe a su vez como parámetros, entre otras, la variable $scope proporcionada por Angular (más adelante se explica para qué se utiliza).
4. Al comienzo del controlador se realiza una petición de un filterManager con RequireJS. Se solicita un acceso a la factoría que se creó anteriormente para las visualizaciones con D3.js y se solicita un objeto del tipo de la visualización (de esta manera se tiene el código más ordenado).
5. Se programa el filtrado que se utiliza para localizar el contenido de los datos dentro de la respuesta que devuelve Elasticsearch, ya que es un objeto con más contenido y solo interesa los datos ofrecidos por las agregaciones. Se guarda en la variable $scope.vis.aggs.bySchemaName['tags'][0] la clave donde se encuentran estos datos dentro del array de objetos.
6. En este paso, se programa la recogida de datos que devuelve Elasticsearch al realizar una consulta. Para realizar esto se dispone de un método especial de Angular llamado $watch, proporcionado por $scope. Este método espera a que haya una modificación de una variable específica que, en este caso, será la proporciona por Kibana para devolver los datos de Elasticsearch (esResponse),

de esta manera se tienen localizados los datos dentro del controlador. En esta función se utiliza el filtro programado anteriormente para localizar la posición del bucket y acceder a los datos.

7. Una vez se tienen los datos cargados, se pueden transferir a un objeto de visualización encargado de mostrarla.

ANEXO B: REQUISITOS NECESARIOS PARA CREAR PLUGINS EN KIBANA

Una forma muy sencilla de tener Kibana funcionando con una versión de Elasticsearch apropiada, es realizando las siguientes acciones:

1. Descargar e instalar Node Package Manager14 (NPM)
2. Clonar el repositorio15 de Kibana en Github.
3. Dentro de la carpeta de Kibana, descargamos las dependencias necesarias ejecutando el comando “npm install”.
4. Descargar la versión de Elasticsearch adecuada para el proyecto ejecutando el comando “npm run elasticsearch” dentro de la carpeta de Kibana. Este comando no solo descarga Elasticsearch, también lo lanza.
5. Dentro de la carpeta de Kibana, instalamos la dependencia de D3 con el comando: “npm install d3v4 --save”.

Una vez Kibana haya lanzado Elasticsearch ya es accesible y se puede comprobar que todo está funcionando correctamente, accediendo a la siguiente dirección con un navegador web (ver figura 10).

[14] https://www.npmjs.com/
[15] https://github.com/elastic/kibana/

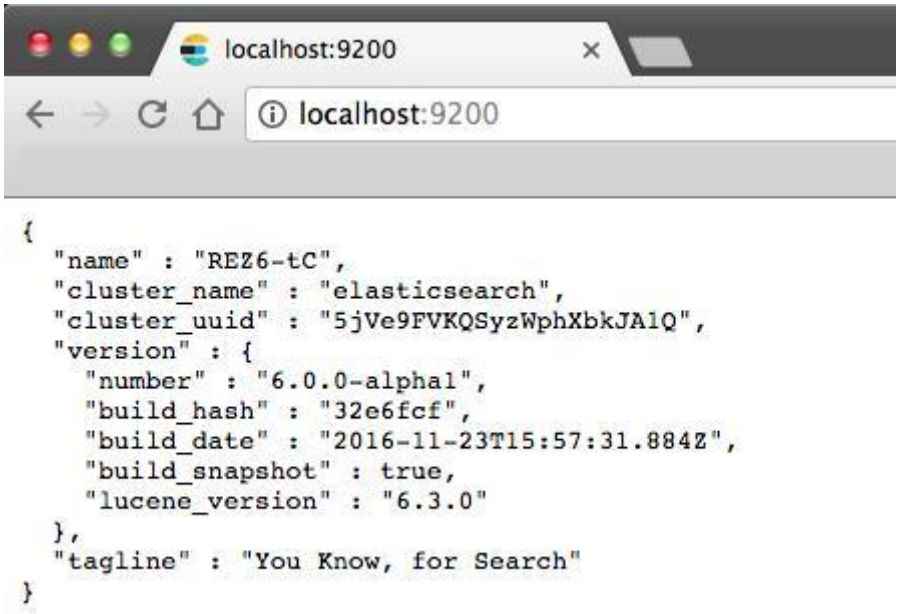

Figura 10 - Respuesta de ElasticSearch una vez instalado y en ejecución

ANEXO C: CÓMO PROBAR UN PLUGIN EN KIBANA

Para probar que todo funciona correctamente, se abre una terminal del sistema y se busca la carpeta del proyecto. Después, una vez se está dentro del proyecto se ejecuta el comando “npm start” (Se da por hecho que ES se está ejecutando antes de lanzarlo).

Después, se introduce en la URL del navegador la siguiente dirección: “https://localhost:5601” y se comienza a iniciar Kibana.

Cuando se abra Kibana, se configura el index creado al comienzo de este proyecto (en este caso el index shakespeare). Para ello hay que seguir los siguientes pasos:

1. Se selecciona la opción “Management” situada en la barra de menú y después “Index Patterns” (ver figura 11).
2. Se da al botón “Add New” en la parte superior (ver figura 12).
3. En la ventana que aparece (“Configure an index pattern”) se introduce el nombre del index y se da al botón “Create” (Ver figura 13).

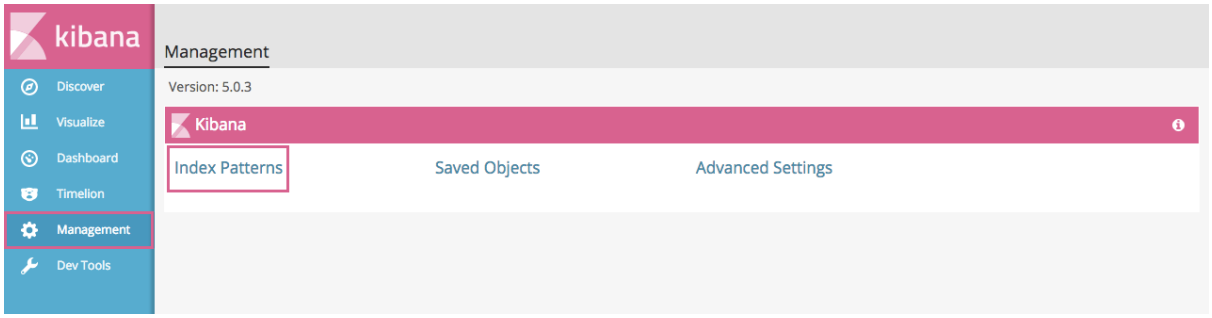

Figura 11. Panel “Management” para insertar un nuevo índice en el panel de Kibana.

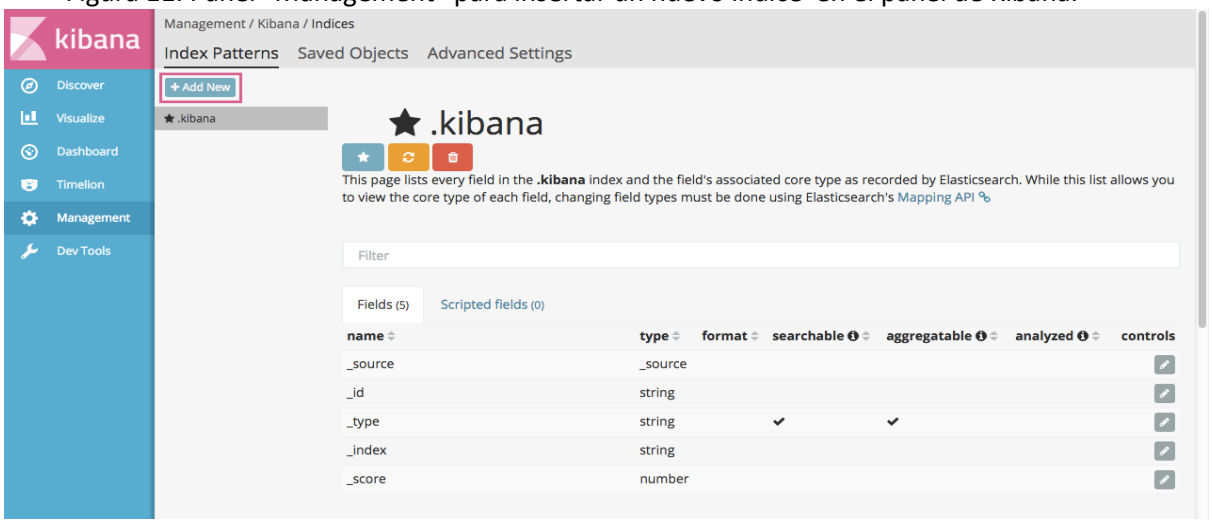

Figura 12. Situación de botón “añadir nuevo” en el panel de administración de índices de Kibana.

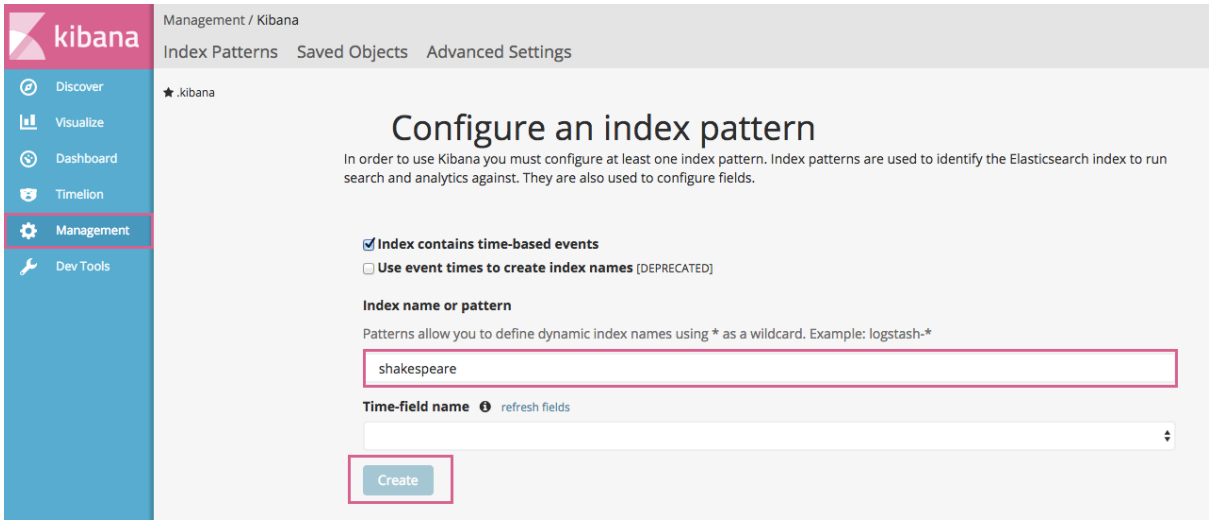

Figura 13. Configuración de un nuevo índice en Kibana.

Una vez se tiene el índice configurado en Kibana se crea una nueva visualización:

1. En el menú de Kibana se selecciona el apartado de visualizaciones (ver figura 14).
2. Buscamos en la lista la visualización que acabamos de crear.
3. Se selecciona el índice que se ha creado en el apartado anterior (ver figura 15).
4. introducimos las métricas y los “buckets” por los que se quiere filtrar la búsqueda de datos y se pulsa el botón de “play” o ejecución (ver figura 16).
5. Si todo ha ido bien debería aparecer en el panel de visualizaciones el diagrama de barras que se ha programado (ver figura 17).

Figura 14. Selección de opción de visualización en el menú de Kibana.

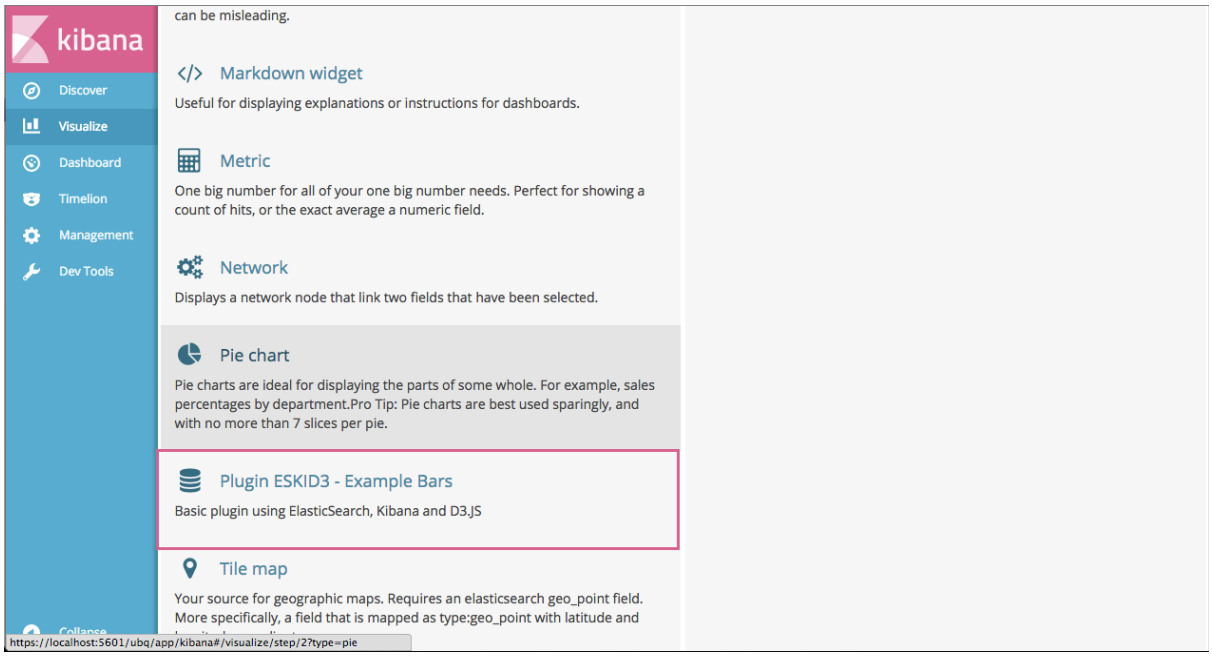

Figura 15. Selección de una visualización en la lista ofrecida por Kibana.

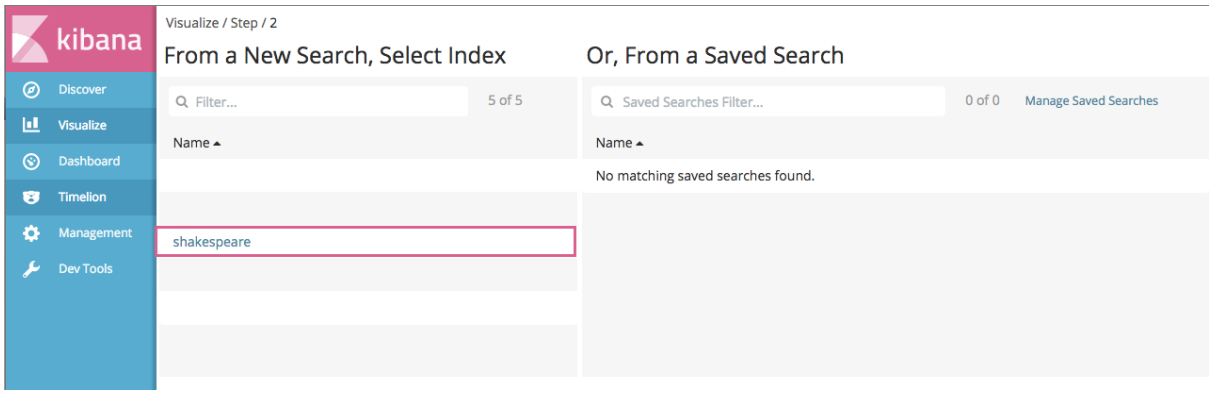

Figura 16. Selección de un índice en el panel de Kibana para asociarlo a una visualización.

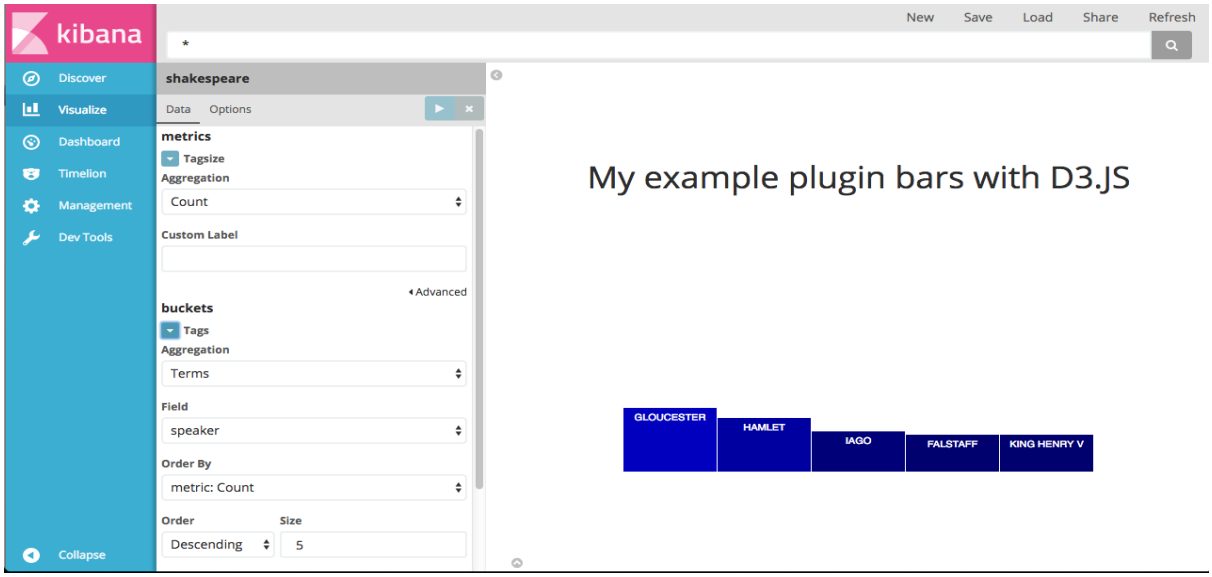

Figura 17. Configuración de métricas y “buckets” dentro de Kibana y visualización de barras.

BIBLIOGRAFÍA

1. Connolly TM, Boyle EA, MacArthur E, Hainey T, Boyle JM. A systematic literature review of empirical evidence on computer games and serious games. Comput Educ. 2012;59(2):661–686.

2. Freire M, Serrano-Laguna Á, Iglesias BM, Martínez-Ortiz I, Moreno-Ger P, Fernández-Manjón B. Game learning analytics: Learning analytics for serious games. Learn Des Technol. 2016;1–29.

3. Domínguez A, Saenz-De-Navarrete J, De-Marcos L, Fernández-Sanz L, Pagés C, MartíNez-HerráIz J-J. Gamifying learning experiences: Practical implications and outcomes. Comput Educ. 2013;63:380–392.

4. Arnab S, Brown K, Clarke S, Dunwell I, Lim T, Suttie N. The development approach of a pedagogically-driven serious game to support Relationship and Sex Education (RSE) within a classroom setting. Comput Educ. 2013;69:15–30.

5. Treviño-Guzmán N, Pomales-García C. How Can a Serious Game Impact Student Motivation and Learning? En: IIE Annual Conference Proceedings. Institute of Industrial Engineers-Publisher; 2014.

6. Laamarti F, Eid M, Saddik AE. An overview of serious games. Int J Comput Games Technol. 2014;2014:11.

7. Serrano-Laguna Á, Martínez-Ortiz I, Haag J, Regan D, Johnson A, Fernández-Manjón B. Applying standards to systematize learning analytics in serious games. Comput Stand Interfaces. 2017;50:116–123.

8. Kiili K, Lainema T, de Freitas S, Arnab S. Flow framework for analyzing the quality of educational games. Entertain Comput. 2014;5(4):367–377.

9. Manero B, Torrente J, Freire M, Fernández-Manjón B. An instrument to build a gamer clustering framework according to gaming preferences and habits. Comput Hum Behav. 2016;62:353–363.

10. Serrano-Laguna Á, Torrente J, Moreno-Ger P, Fernández-Manjón B. Application of learning analytics in educational videogames. Entertain Comput. 2014;5(4):313–322.

11. Freire M, Manjón BF. D21 Metodología de Integración de Objetos de Aprendizaje Avanzados en Sistemas de E-Learning. [último acceso 12 de abril de 2017]; Disponible en: http://www.e-ucm.es/drafts/e-UCM_draft_300.pdf

12. Kuznetsov SD, Poskonin AV. NoSQL data management systems. Program Comput Softw. 1 de noviembre de 2014;40(6):323-32.

13. Alonso Fernández C, Alonso Fernández C. Gaming learning analytics for serious games [Internet]. 2016 [último acceso 15 de abril de 2017]. Disponible en: http://eprints.ucm.es/38711/