StopFlu: Sistema BI d'anàlisi predictiu contra la grip

Ruben Vidal Almerge

Treball de final de Màster

Màster en Enginyeria Informàtica

Consultor: Felipe Geva Urbano

Curs 2013/2014 - Segon quadrimestre

FITXA DEL TREBALL FINAL

Títol del treball:	*StopFlu: Sistema BI d'anàlisi predictiu contra la grip*
Nom de l'autor:	*Ruben Vidal Almerge*
Nom del consultor:	*Felipe Geva Urbano*
Data de lliurament (mm/aaaa):	*06/2014*
Àrea del Treball Final:	*Business Intelligence*
Titulació:	*Màster en Enginyeria Informàtica*

Resum del Treball (màxim 250 paraules):

Vivim, cada cop més, en un món tecnològic, on la vida diària es comparteix a les xarxes socials quasi sense adonar-nos-en. En aquest context, es generen quantitats ingents d'informació que, un cop tractades, poden ésser útils en estudis ben diversos com són la detecció de terratrèmols o la detecció prematura d'una epidèmia. En relació a aquest últim, el virus de la grip és un greu problema de salut pública ja que es destinen part dels recursos sanitaris durant un període de temps considerable i disminueix la productivitat laboral dels afectats que la pateixen. Davant d'aquesta situació, es planteja la realització d'un sistema de *Business Intelligence* que analitzi les dades extretes dels *tweets* de la plataforma *Twitter* en relació a les hospitalitzacions produïdes a un hospital de Catalunya, per tal de tenir un anàlisi predictiu de l'aparició d'un brot d'aquestes característiques. El treball va més enllà al emprar una tecnologia no convencional per la implementació del sistema BI. S'escull la dupla *Elasticsearch* i *Kibana* per tal d'aconseguir un sistema robust, distribuït, escalable i, sobretot, totalment personalitzable. Després d'un estudi

d'aquestes dos solucions, incloent els *plugins* de monitoratge i càrrega de dades, s'ha elaborat un *data warehouse* complet i un quadre de comandament introductori. Es deixa, per futures línies de treball, l'anàlisi profund de les dades i la conseqüent extracció d'uns resultats que ens ajudin a predir amb una major antelació l'aparició d'un nou brot del virus de la grip.

Abstract (in English, 250 words or less):

We are living in a technological world and, more and more, our daily life is shared through social media almost without realising it. In this context, huge amount of data are generated which, once treated, may be useful for many and diverse studies such as earthquake detection or early epidemic detection. Regarding this last example, influenza virus is a serious problem for public health because of the several healthcare resources allocated in a short time period and because it decreases labour productivity of patients suffering it. Given this situation, the creation of a Business Intelligence System which could give a predictive analysis of flu outbreak is considered, using data obtained from tweets of Twitters users related to hospitalization in Catalan hospitals. This project goes one step further by using a non conventional technology for the BI system implementation. Elasticsearch and Kibana are chosen to obtain a robust, distributed, scalable, and particularly a fully customizable system. After studying both solutions, including the site and data load plug-ins, a complete data warehouse and an initial dashboard were developed. Deep analysis of data and subsequent results extraction which could help to better predict in advance the emergence of a new influenza virus outbreak is left for future works.

Paraules clau (entre 4 i 8):

Grip*, Twitter, Elasticsearch, Kibana, Marvel, plugin i river.*

Índex

Llista de figures

1. Introducció

1.1 Context i justificació del Treball

Entre les malalties infeccioses més comunes i amb major impacte a la població en països desenvolupats, es troba el virus de la grip. Aquesta malaltia afecta a les vies respiratòries i els seus símptomes inicials són similars als d'un refredat comú.

La grip es transmet des d'individus infectats a través de partícules a l'aire carregades de virus, que són emeses per la tos, els esternuts o la parla. També és transmissible per la sang i per les superfícies o objectes contaminats amb el virus.

Per reduir els casos de grip, als països desenvolupats s'han establert campanyes de vacunació anual per a les persones que tenen un major risc de contraure la malaltia o que són més vulnerables a les complicacions derivades de la mateixa. No obstant, a causa de què la vacuna no és obligatòria i part de la població no es vacuna, desenes de milers de persones a Catalunya contrauen el virus de la grip cada any. La majoria de casos d'infecció es donen al període hivernal, quan el virus és més resistent. Per altra banda, predir amb exactitud la data exacta de màxima efervescència del virus és prou complex, i normalment es descobreix a posteriori, un cop ja es té l'evidència d'un gran nombre de contagis.

El virus de la grip és un greu problema de salut pública ja que ocupa part dels recursos sanitaris durant un període de temps considerable i té importants repercussions econòmiques, no només per les despeses sanitàries que provoca, sinó degut a la reducció de la productivitat laboral dels afectats. D'aquí sorgeix la necessitat d'utilitzar noves tècniques capaces de predir amb una major antelació l'aparició d'un nou brot.

1.2 Objectius del Treball

L'objectiu d'aquest Treball de Final de Màster és generar un sistema predictiu que permeti estimar brots gripals abans que es produeixin i integrar aquest sistema als sistemes d'intel·ligència de negoci (BI) de l'empresa (que podria ésser l'Institut Català de la Salut, per exemple).

Dintre d'aquest context , el treball es divideix en 4 punts:

- **Recuperació de la informació**: Amb l'ajuda de la plataforma *Twitter*, es recopilaran tots els missatges relacionats amb la malaltia mitjançant certes paraules clau, com poden ser: grip, grip A, malaltia, llit, etc. Les cerques seran geolocalitzades ja que només interessen els missatges de Catalunya i els voltants.

- **Càrrega de dades**: Disseny d'un magatzem de dades (*data warehouse*) per tal de guardar les dades recuperades de *Twitter*. La proposta de treball ja facilita les dades d'hospitalització i d'urgències generades a un centre hospitalari referents als 2 anys anteriors (2012 i 2013). Aquestes últimes també s'incorporaran al sistema descrit.

- **Creació d'un quadre de comandament**: Amb l'ajuda d'un sistema d'indicadors (KPI), confecció d'un quadre de comandament on es reflecteixi clarament l'impacte actual de la grip a *Twitter* en altres períodes passats que hagin precedit a un brot del virus de la grip.

- **Explotació de les dades**: El sistema ha de respondre a les necessitats de l'usuari final (interacció usuari-quadre de comandament) i arribar a plantejar possibles accions futures, com la comparació amb uns nous KPIs, l'ús d'altres sistemes d'anàlisi o l'ús d'altres fonts de dades en la predicció.

1.3 Enfocament i mètode seguit

Tractant-se d'un exercici eminentment d'intel·ligència de negoci, l'opció més lògica seria la utilització d'un producte ja establert al mercat i adaptat als fluxos de treball d'aquesta matèria. Però es planteja el repte d'anar una mica més enllà i muntar un sistema propi partint des de zero. Després de valorar diferents opcions, s'escull un sistema d'emmagatzematge i cerca relativament nou, que aporta una gran escalabilitat i una flexibilitat d'ús molt potent. D'aquesta manera es vol cobrir el fenomen del *Big Data*, és a dir, la gestió d'un gran volum de dades per tal d'avaluar o analitzar patrons quotidians d'una empresa. Per tant, es basa l'estratègia del projecte en la combinació de dos productes joves però amb un gran potencial: *Elasticsearch* i *Kibana*, que s'analitzen més endavant.

1.4 Planificació del Treball

En aquest punt es fa un anàlisi de cadascuna de les tasques que s'han de realitzar, tot consultant el calendari del quadrimestre (laboral i entregues de les PACs), per tal d'acabar confeccionant una planificació i marcant unes fites que s'hauran de complir.

Les tasques es defineixen en funció dels diferents lliuraments que marca el pla docent del treball. Es busca el paral·lelisme amb un projecte 'real', on el client vol anar seguint el desenvolupament de la feina feta i veure si es compleixen els acords establerts a l'inici del projecte. Dins de cada lliurament, s'avaluarà la càrrega de feina que serà possible desenvolupar.

La següent taula resum serveix per ordenar les tasques a realitzar:

Tasca	Precedents/ observacions
1. Exercici d'anàlisi previ per tal saber la temàtica del meu Treball de Final de màster. Redacció de diferents correus electrònics als responsables de les àrees demanant informació sobre les propostes ofertes.	
2. Presa de decisió sobre l'àrea del meu TFM. Després de l'ajuda dels consultors i del tutor, es pren la decisió d'enfocar el meu TFM en l'àrea de BI.	Punt 1
3. Cerca de documentació general de la matèria. Acabo trobant a la biblioteca de la UOC el llibre 'Introducció al Business Intelligence'.	Punt 1 i 2
4. Lectura i acceptació de la proposta de TFM: 'Sistema de BI para la previsión de brotes gripales a partir de información de redes sociales'.	
5. Videoconferència amb els professors assignats i el company que també decideix acceptar la proposta comentada anteriorment.	
6. Visió general de la metodologia de treball a desenvolupar durant tot el projecte. Realització de diverses proves per veure si el *data warehouse* + motor de cerca (*Elasticsearch*) i el quadre de comandament (*Kibana*) escollits poden resoldre les exigències demanades pel cas que ens ocupa.	El punt 5 serveix de punt de partida i treu diversos dubtes sobre la direcció escollida pel sistema BI.
7. Confecció del pla de treball (PAC 1)	S'entrega amb uns dies de retard a causa d'una incidència laboral
8. Anàlisi de les dades d'hospitalització i d'urgències facilitades a l'enunciat del treball, per tal de trobar les dades necessàries i començar a definir indicadors.	

9. Sèrie de proves per l'extracció de dades de la plataforma '*Twitter*', tot seguint les notes tècniques facilitades al punt 4.3 de l'enunciat (*API de Twitter*)	
10. Estudi de les aplicacions necessàries per la implementació del sistema.	Punt 6
11. Estat de l'art, Anàlisi i Disseny (PAC 2)	Data proposada: 14/4/2014
12. Creació del quadre de comandament. S'utilitza la dupla formada per *Elasticsearch* i *Kibana*, que donaran un efecte visual important i un rendiment molt optimitzat, afavorint l'escalabilitat amb la inclusió de nodes al sistema.	
13. Càrrega de les dades al sistema implementat	
14. Explotació de les dades, on es vol treure el màxim profit al sistema plantejat i que pugui ajudar en la tasca encomanada: la predicció de brots de grip.	Imprescindible un bon punt 12
15. Implementació i Memòria Preliminar (PAC 3)	Data proposada: 19/5/2014
16. Plantejament d'accions futures. Comparació amb nous KPIs, un ús d'altres sistemes d'anàlisis o un ús d'altres fonts de dades en la predicció.	Tots els punts anteriors han d'estar ben finalitzats
17. Entrega Final (PAC 4). Lliurament de la memòria i la presentació.	Punts anteriors ben resolts. Data Fixa: 10/6/2014
18. Debat virtual: entre els dies 18 i 20 de juny, el tribunal plantejarà preguntes i situacions a l'alumne per mitjà del correu electrònic. Aquests missatges s'hauran de contestar en el termini de 24 hores.	

1.5 Breu sumari de productes obtinguts

L'objectiu principal es centra en trobar un sistema fiable de BI, recolzat per un *data warehouse* escalable i distribuït, amb un quadre de comandament realitzat amb *Kibana* que reflecteixi els resultats buscats i pugui servir en un entorn real per usuaris de tots els nivells.

1.6 Breu descripció dels altres capítols de la memòria

Seguint amb la planificació i els objectius fixats, la memòria d'aquest projecte tindrà els següents apartats:

- Capítol 2 - '*Twitter*: Més que una xarxa social': Breu explicació de la situació actual pel que fa a la utilització de les noves tecnologies i les repercussions que poden esdevenir amb un estudi estructurat de tota la informació que es genera.

- Capítol 3 - Disseny del sistema BI: Distribuït en diferents subapartats, es realitzarà un anàlisi complet de les eines escollides per tal d'implementar el sistema de *Business Intelligence* perseguit. A continuació, es seguiran els punts plantejats com a objectius.

- Capítol 4 - Explotació de les dades: Mitjançant el quadre de comandament, s'efectuarà un estudi de les sinergies que poden existir entre la plataforma *Twitter* i les dades recollides sobre el virus de la grip.

- Capítol 5 - Conclusions: Inclourà una descripció de les conclusions del treball, una reflexió crítica sobre l'assoliment dels objectius formulats inicialment i un anàlisi crític del seguiment de la planificació i metodologia al llarg del producte. Finalment, es plantejaran les línies de treball futur que no s'han pogut explorar en aquest treball i han quedat pendents.

- Capítol 6 - Glossari: Definició dels termes i acrònims més rellevants utilitzats dins la Memòria

- Capítol 7 - Bibliografia: Recull de tota la documentació consultada durant l'elaboració del projecte. Es classificarà en tres grups: Llibres, articles científics i referències Web.

- Capítol 8 - Annex 1: Detall dels fitxers de configuració emprats en la parametrització dels productes *Elasticsearch* i *Kibana*.

2. *Twitter*: Més que una xarxa social

La consolidació de les tecnologies de consum a les vides de la majoria d'éssers humans és ja tota una realitat i existeix el fenomen de no poder viure sense estar connectat. En aquest sentit, els usuaris de les diferents xarxes socials expressen tot tipus d'informació sobre aspectes diaris de la seva vida. Així, la plataforma *Twitter*, un popular servei de *microblogging*, pot esdevenir en una bona font potencial per a la detecció del virus de la grip, i d'altres malalties, en els seus primers estadis, en gran part pel seu funcionament en temps real.

Quan el virus de la grip comença a propagar-se, els usuaris actius de *Twitter* infectats acostumen a publicar *tweets* ('piulades', comentaris de 140 caràcters al seu mur) relacionats amb el seu estat actual. És d'ús comú trobar missatges com el següent: "Tancat a casa amb grip" o "No puc anar a treballar, la grip em té al llit", per exemple. Aquest fet, aparentment insignificant i intranscendent, pot arribar a considerar-se com un indicador d'un contagi massiu del virus i permetre la detecció de l'epidèmia de la grip ràpidament. Gràcies a la immediatesa de la xarxa, s'ha utilitzat *Twitter* per moltes aplicacions en temps real com són el seguiment de la detecció de terratrèmols, seguiment de diferents afectacions de la salut pública i, per què no, pot ésser molt útil en la detecció prematura d'una epidèmia tan comú com el virus de la grip.

3. Disseny del sistema BI

3.1 Decisions preses

Com s'ha comentat anteriorment, aquest treball està enfocat a profunditzar en l'estudi d'unes eines relativament noves, però que estan agafant prou rellevància gràcies a la seva potència de processament i la fàcil configuració i desplegament en tota mena d'entorns i aplicacions derivades. Pels dos primers punts del projecte, la recuperació de la informació i la càrrega de dades, s'utilitza un motor de cerca distribuït i *open source* anomenat *Elasticsearch* [1]. L'enginyer *Shay Banon* es va adonar que necessitava crear una solució de cerca escalable arrel de la programació de la tercera versió del seu *Framework* anomenat *Compass* [2]. Així es decidí per implementar una solució integrada des de la base per tal d'ésser distribuïda, amb una interfície comú, amb un format JSON [3] sobre el protocol HTTP, adequat pels llenguatges de programació que no fossin de Java. Va llançar la primera versió de *Elasticsearch* al febrer de l'any 2010 i actualment, després de diverses actualitzacions, va per la versió 1.1.1 (abril-maig del 2014), la qual utilitzo pel meu sistema.

Dintre del paquet d'aplicacions que s'ofereixen a la pàgina web de *Elasticsearch*, en destaca un producte també de codi obert, amb una analítica basada en el navegador i una interfície gràfica molt intuïtiva, que pot encaixar perfectament en els dos últims punts del meu treball, que són l'elaboració d'un quadre de comandament i l'explotació de dades, per tal d'obtenir un sistema BI predictiu ajustat a les necessitats del treball. El producte en qüestió s'anomena *Kibana* [4].

Intentant donar una visió global del sistema, s'enumeren les principals característiques d'aquest potent motor de cerques, on es constata la potència inherent de la tecnologia desenvolupada:

- Distribuït: *Elasticsearch* està preparat per tal d'escalar horitzontalment. Es pot començar amb una solució petita i anar creixent a mesura que el negoci ho necessita. En aquest projecte es treballa amb poques dades, però es vol configurar un sistema que sigui fàcilment configurable per tal d'afegir nodes i processar un gran volum de dades (*Big data*) quan sigui necessari. Del funcionament intern de la plataforma es parla en el següent punt.

- Alta disponibilitat: els clústers de *Elasticsearch* són flexibles. Gràcies a un sistema de localització i reconeixement de nodes, el sistema és capaç de detectar i eliminar nodes que estan fallant, reorganitzant-se per tal d'assegurar que les dades estan salvades i segueixen accessibles.

- Dades en temps real: Les dades a una aplicació estan en constant canvi. *Elasticsearch* permet disposar dels últims canvis realitzats sobre les dades de manera immediata.

- *Multi-tenancy* (tinença múltiple): Un clúster de *Elasticsearch* pot allotjar múltiples índexs, que poden ésser consultats de manera independent o en grup. A més, assignant àlies als índexs, permet afegir índexs al vol, de manera transparent, per l'aplicació.

- Cerques *full-text*: *Elasticsearch* es basa en *Lucene* [5] per tal de proveir, amb les millors capacitats, la cerca de text. Les cerques suporten múltiples idiomes, la geolocalització, l'auto-completat, etcètera.

- Orientat a documents: Les entitats s'emmagatzemen com a documents JSON estructurats. Tots els camps són indexats per defecte i tots els índexs poden ser utilitzats a una mateixa consulta.

- Gestió de conflictes: Té uns mecanismes (*optimistic version control*) [6] per tal d'assegurar que les dades mai es perden per culpa de canvis simultanis sobre un mateix document, realitzats per diferents processos.

- Sense esquemes: *Elasticsearch* permet treballar sense esquemes que defineixin l'estructura de les dades. Simplement passant-li un document JSON, intentarà detectar l'estructura de les dades, indexar-les i deixar-les accessibles per a les cerques.

- API *Restful*: Proporciona una API (Interfície de programació d'aplicacions) de tipus *Restful* [7], amb JSON i sobre el protocol HTTP, que permet realitzar quasi qualsevol acció. També existeixen APIs per d'altres llenguatges, entre els que es troba el famós i estès *Java*.

- Persistència a nivell d'operació: *Elasticsearch* vetlla per la seguretat de les dades. Tots els canvis que es realitzen sobre els documents s'emmagatzemen en *logs* de transaccions en múltiples nodes del clúster, per tal de minimitzar les pèrdues d'informació.

- *Open Source*: Es tracta d'un software de codi lliure i està sota la llicència Apache2 [8]. Permet descarregar, utilitzar i modificar el codi, característica fonamentalment per poder-se desmarcar de productes comercials que no s'ajustin completament a les necessitats buscades.

Finalment, la inclusió de *Kibana* dintre del sistema BI es realitza mitjançant un enllaç simple de la configuració de la mateixa per tal que utilitzi les dades emmagatzemades al motor de cerca. Es detalla a l'apartat de la implementació del quadre de comandament.

3.2 Anàlisi tècnic de *Elasticsearch*

La principal diferència d'aquest producte, enfront d'altres existents, és el fet de poder explorar i utilitzar les dades lliurement, amb un mètode transparent i fiable d'emmagatzematge i cerca de les mateixes. L'arquitectura de l'aplicació no és una 'caixa negra', com succeeix en la majoria de productes BI, sinó que es pot consultar i gestionar segons les necessitats de l'usuari o l'administrador.

En aquest treball, s'expliquen a grans trets els conceptes de clústers i nodes, per tal d'entendre la funcionalitat distribuïda i escalable d' *Elasticsearch*.

Un node és una instància en execució de *Elasticsearch*. Un clúster és un grup de nodes amb un mateix nom, que estan treballant junts per tal de compartir les dades, proporcionar commutació per possibles fallades i garantir l'escalat, encara que un únic node pot formar un clúster per si mateix. D'aquesta manera, es pot escalar l'escenari a centenars (o inclús milers) de servidors que actuen com a nodes i gestionar *petabytes* [9] de dades, si fos necessari.

Encara que *Elasticsearch* és distribuït per naturalesa i està dissenyat per tal d'ocultar la complexitat associada, aquestes són algunes de les operacions que succeeixen de forma automàtica:

- Partició dels documents en diferents recipients o fragments, que poden emmagatzemar-se en un únic node o en diferents nodes.
- L'equilibri d'aquests fragments mitjançant els nodes al clúster, per difondre la indexació i la cerca de la càrrega.
- Duplicació de cada fragment per tal de proporcionar còpies redundants de les dades, per evitar la pèrdua de dades en cas de fallada de *hardware*.
- Sol·licituds d'encaminament des de qualsevol node del clúster als nodes que contenen les dades que es volen mantenir.
- La perfecta integració de nous nodes al clúster, tant si s'augmenta com si es disminueix el nombre. En aquest últim cas, es produeix una redistribució dels fragments per tal de recuperar-se de la pèrdua del node.

Elasticsearch està dissenyat per estar sempre disponible i l'escalat hi juga un paper molt important. Aquest pot venir donat per la compra de servidors més grans (escalat vertical o ampliació) o per la compra de més servidors (escalat horitzontal). L'escalat real prové de l'escalat horitzontal –la possibilitat

d'afegir més nodes al clúster i poder balancejar la càrrega entre ells. A la majoria de les bases de dades relacionals, l'escalat horitzontal necessita d'una important remodelació per tal d'assimilar la nova arquitectura. En canvi, *Elasticsearch* es distribueix automàticament, per tal de treballar amb múltiples nodes i proporcionar una bon escalat i una alta disponibilitat.

Si començo amb un sòl node, sense dades ni índexs, el clúster es presenta de la següent manera:

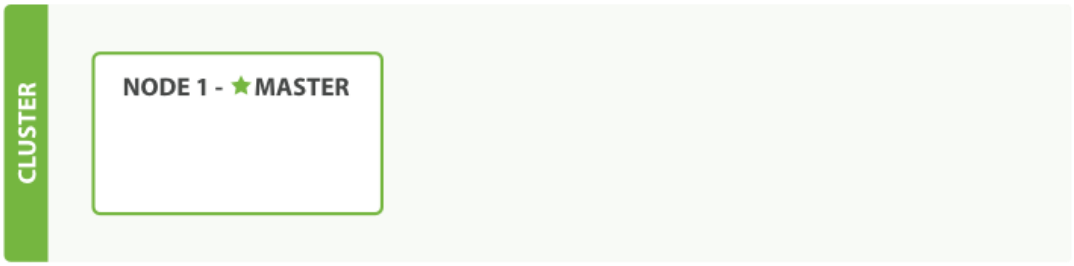

Figura 1 - Clúster amb un sòl node

El node s'escull com a mestre (*master*), que és l'encarregat de gestionar els canvis de tot el clúster, com poden ser la creació o eliminació d'un índex, o la suma o resta d'un node al clúster. El node mestre no necessita estar implicat en els canvis a nivell de document o cerques, el que significa que tenir un sol node mestre no es converteixi en un coll d'ampolla a mesura que creix el tràfic al sistema. Qualsevol node pot esdevenir en el mestre si és necessari.

Per afegir dades a *Elasticsearch*, necessito un índex –un lloc per emmagatzemar dades relacionades. Un índex és només un *logical namespace* [10] que apunta a un o més fragments físics.

Un fragment és una 'unitat de treball' de baix nivell. Cada fragment és una instància única de *Lucene* i un complet motor de cerca en si mateix. Els documents són emmagatzemats i indexats en fragments, però les aplicacions no parlen directament amb ells. Parlen amb un índex.

Els fragments es distribueixen per tots els nodes del clúster. S'ha de pensar en els fragments com a contenidors de dades. Els documents s'emmagatzemen en fragments i aquests són assignats als nodes del clúster. A mesura que el grup creix o es redueix, *Elasticsearch* migra automàticament els fragments entre els nodes per a què el grup es mantingui balancejat. Un fragment pot ser primari o una rèplica del mateix. Cada document a l'índex pertany a un sòl fragment primari, de manera que el nombre de fragments primaris es determina per la quantitat màxima de dades que l'índex pot contenir. Encara que no hi ha un límit teòric per la quantitat de dades que un fragment primari pot contenir, sí que existeix un límit pràctic, que vindrà donat pel *hardware* del sistema, la mida i la complexitat dels documents, com s'indexen i es consulten els documents, i si s'obtenen uns temps de resposta acceptables.

Un fragment de rèplica és una còpia d'un fragment primari. Les rèpliques s'utilitzen per tal de proporcionar còpies redundants de les dades per protegir-se contra fallades de hardware, i per servir les peticions de lectura, com són la cerca o la recuperació d'un document.

Un cop es crea un índex, es presenta la següent estructura:

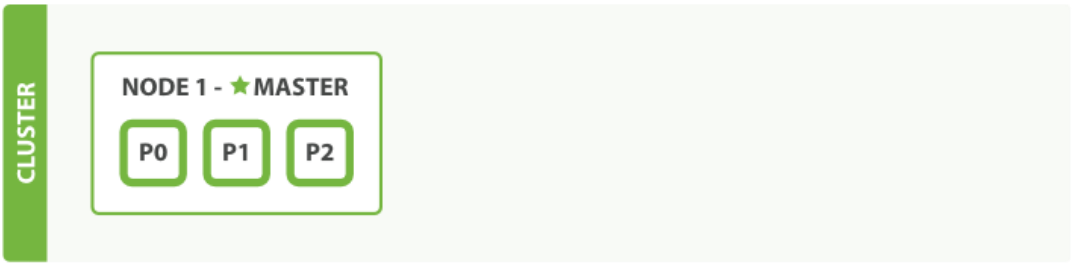

Figura 2 - Clúster amb un node i un índex

L'execució d'un sòl node significa que es té un únic punt de fallada perquè no existeix redundància a cap nivell. Només cal afegir un segon node, ja sigui local o remot, per tal de protegir el sistema contra la pèrdua de dades. El nou node s'uneix al grup de manera automàtica, sempre i quan tingui el

mateix nom de l'agrupació al fitxer de configuració, establint així una comunicació amb els altres nodes. L'esquema queda com segueix:

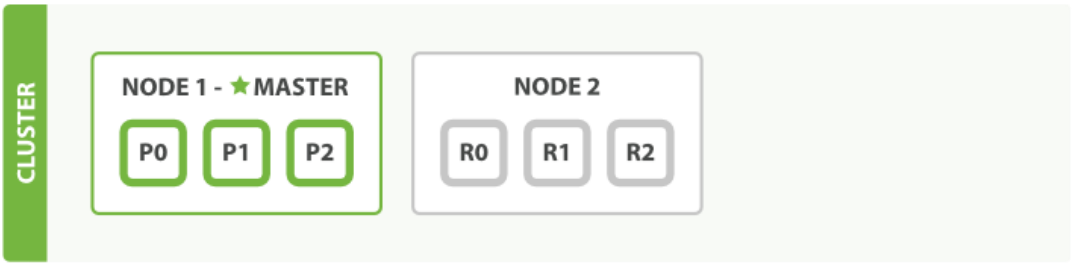

Figura 3 - Clúster amb dos nodes

El segon node s'ha afegit a l'agrupació i 3 fragments de rèplica s'han assignat a la mateixa, una per a cada fragment primari. Ara es podria perdre qualsevol dels 2 nodes i totes les dades quedarien intactes.

Qualsevol document indexat per primer cop s'emmagatzema en un fragment primari, per ésser copiat en paral·lel cap al fragment de rèplica associat. Això assegura que el document es pugui obtenir a partir d'un fragment primari o de qualsevol de les seves rèpliques.

Finalment, amb la inclusió d'un tercer node, es produeix un escalat horitzontal com el de la figura:

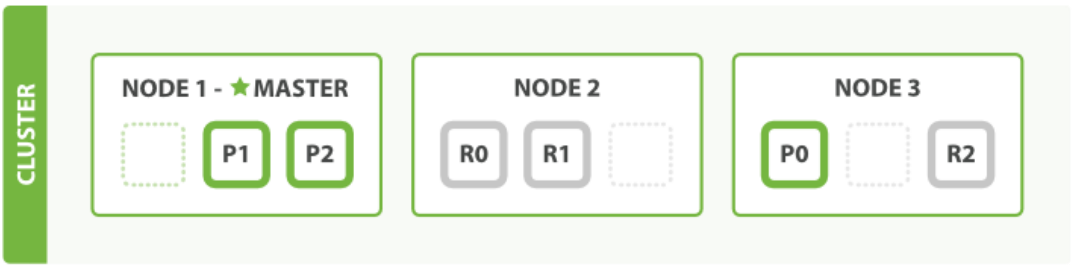

Figura 4 - Inclusió d'un tercer node

Un fragment de cada node 1 i node 2 es trasllada al nou node 3 i es tenen 2 fragments per node, enlloc de tres. Això significa que els recursos de hardware (CPU, RAM, E/S) de cada node es comparteixen entre un nombre

menor de fragments, permetent que cada fragment treballi en millors condicions.

Un fragment és un motor de cerca per si mateix i és capaç d'utilitzar tots els recursos d'un únic node. Amb tots els fragments (3 primaris i 3 rèpliques), l'índex és capaç d'escalar a un màxim de 6 nodes, amb un fragment a cada node i que cada fragment tingui accés al 100% dels recursos del seu node.

Què passa si augmento el nombre de rèpliques a 2? L'índex en qüestió tindrà ara 9 fragments: 3 primaris i 6 rèpliques. Si tingués que afegir uns altres 3 nodes al clúster, tindria de nou un fragment per node, i l'agrupació seria capaç de manegar un 50% més de sol·licituds que abans.

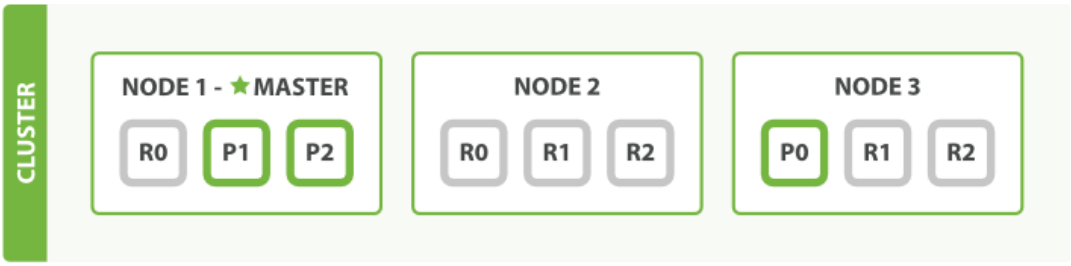

Figura 5 - Rèpliques

Cal destacar que tenir més fragments de rèplica en el mateix nombre de nodes no augmenta el rendiment en absolut, perquè cada fragment té accés a una petita fracció dels recursos del seu node. És necessari, doncs, afegir *hardware* per tal d'augmentar el rendiment. Però amb aquestes rèpliques addicionals no significa que es tingui més redundància. Amb la configuració del node anterior, ara es poden perdre 2 nodes sense que esdevingui cap pèrdua de dades.

En el cas que caigui el node número 1, que a més és el mestre, el clúster canvia automàticament de node mestre i escull al node 2. El fragments primaris 1 i 2 es perden quan cau el node 1, i l'índex no pot funcionar correctament si no troba els fragments primaris. Un senyal d'alerta al clúster marca en vermell que no tots els fragments primaris estan actius.

Afortunadament existeix una còpia completa dels 2 fragments primaris perduts en els altres nodes, així que el primer que succeeix és que el nou node mestre promou el canvi dels fragments rèplica a primaris, per tal d'assegurar la integritat de les dades i tornar l'estabilitat al sistema (Figura 6).

Figura 6 - Fallada de node 1

3.3 Funcionalitats extra: *plugins* i *rivers*

El clúster d'un sistema *Elasticsearch* és autosuficient i autogestionable, però precisament un dels avantatges de treballar amb aquesta arquitectura és el fet de disposar d'un control complet de l'estat del sistema en cada moment. Amb simples crides es podria arribar a obtenir tota una sèrie d'estadístiques que donarien una visió global del sistema. Gràcies als desenvolupadors propis i una comunitat creixent al darrera, existeixen cada dia més *plugins* per facilitar la gestió de *Elasticsearch*.

Els *plugins* són una forma de millorar la funcionalitat bàsica del sistema d'una manera personalitzada. Van des d'analitzadors, *scripts* natius, funcions de descobriment de nodes i més.

La instal·lació dels mateixos és prou simple ja que la majoria es poden trobar al *GitHub* [11] de *Elasticsearch* i des de la carpeta '/bin' es pot executar la següent comanda:

```
Plugin - instalar <org> / < usuario / componente > / < versión >
```

Es descarreguen automàticament del repositori, encara que també es pot utilitzar l'adreça URL que correspongui.

Uns dels més importants són els corresponents a l'estabilitat (*site*) del sistema. Aquests aporten informació sobre l'estat actual del clúster i permeten detectar ràpidament qualsevol anomalia o problema en l'estructura de les dades. S'analitzen els 3 més rellevants desenvolupats per la comunitat d'usuaris:

- **HEAD** [12]: Dóna una visió general del clúster, tot mostrant la topologia del mateix i permetent realitzar operacions d'índex i nivell de node. A més, ofereix un parell d'interfícies de cerca que permeten consultar resultats en format JSON i crides arbitràries a la API REST.

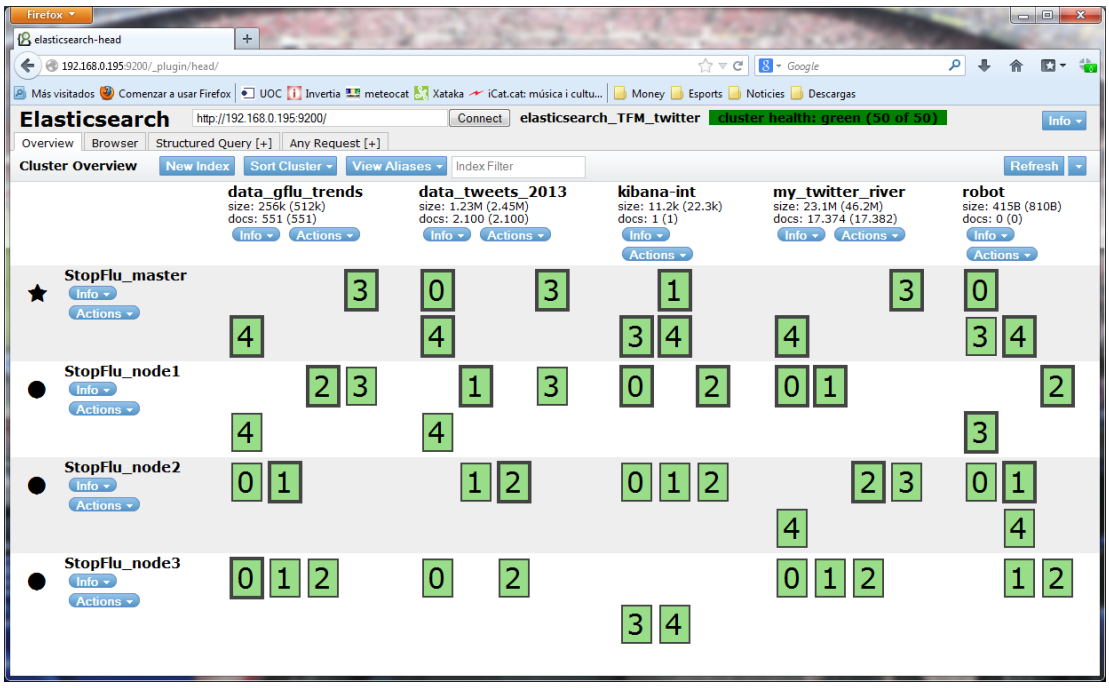

Figura 7 - Plugin Elasticsearch-head

Gràcies a aquesta imatge del *plugin* (figura 7), es pot veure tota l'estructura de dades explicada en l'apartat anterior. El clúster disposa de 4 nodes (amb un mestre, òbviament), 5 fragments per cada node i 2 rèpliques de cadascun d'ells. S'aprecia perfectament que la distribució està balancejada i l'estat del clúster és verd (indicador de bona salut del

sistema). Si s'atura manualment un dels nodes, l'escenari canvia a la següent imatge del sistema:

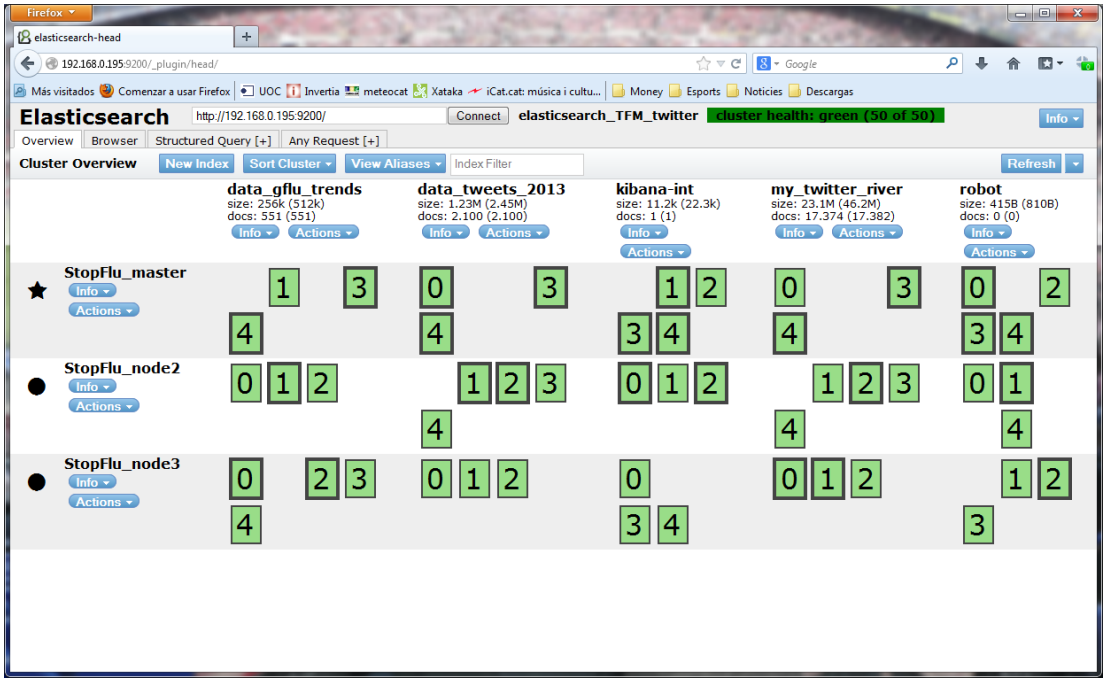

Figura 8 - Fallada del node 1 al clúster

La fallada del node 1 obliga al sistema a redistribuir els fragments i les rèpliques per tal d'assegurar la integritat de les dades. Automàticament els fragments que s'emmagatzemaven al node caigut són allotjats en diferents nodes, gràcies a les rèpliques, i tornant a assignar fragments primaris (són els quadrats marcats amb un contorn més fort).

Un cop vista la topologia del clúster, es poden analitzar les dades mitjançant el següent apartat disponible a la plana principal del *plugin*. Si es selecciona la pestanya anomenada *browser*, s'accedeix a un navegador que aglomera, de forma prou clara i intuïtiva, tota la informació resident al sistema (Figura 9).

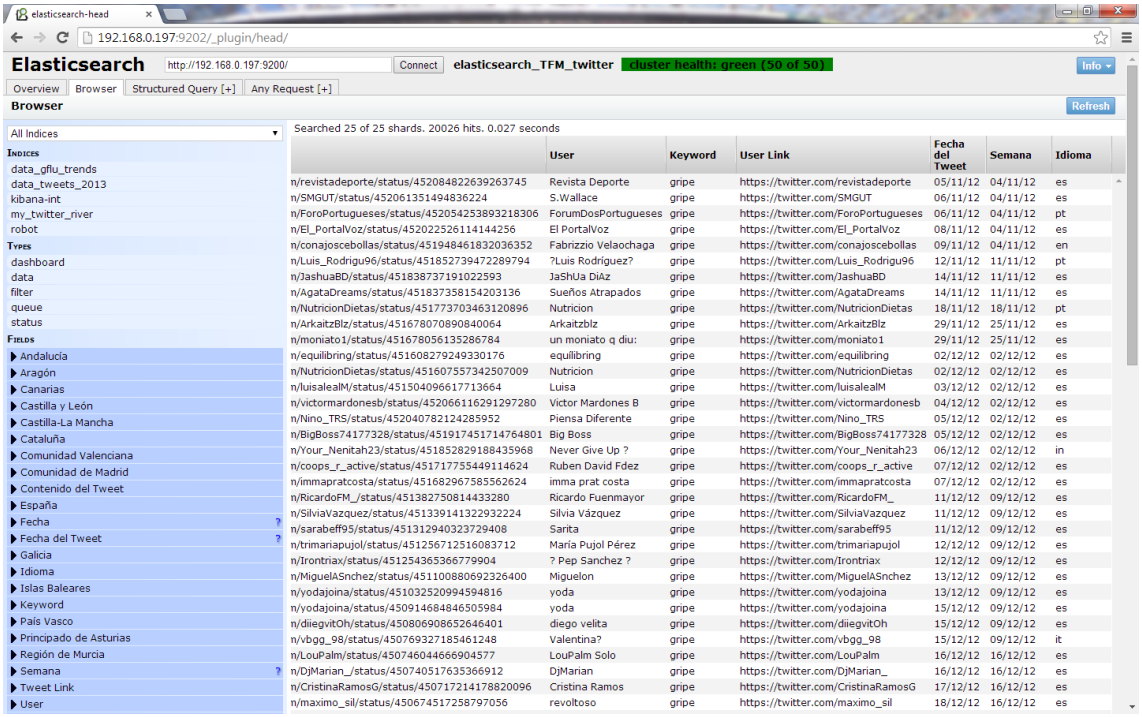

Figura 9 - Navegador del plugin

En aquest imatge s'observa perfectament l'estructura de les dades emmagatzemades als diferents índexs, que a la vegada són de tipus diferents i cadascun disposa dels seus camps corresponents. Es poden realitzar cerques d'una manera molt simple, seleccionant l'índex, el tipus i, inclús, una característica d'algun dels camps de la base de dades. Si per contra es necessita fer una cerca estructurada, en la següent pestanya podem efectuar una consulta més refinada.

En la següent imatge es pot veure el resultat de consultar els *tweets* escrits en espanyol i recollits a la primera setmana de desembre de l'any 2012 (Figura 10).

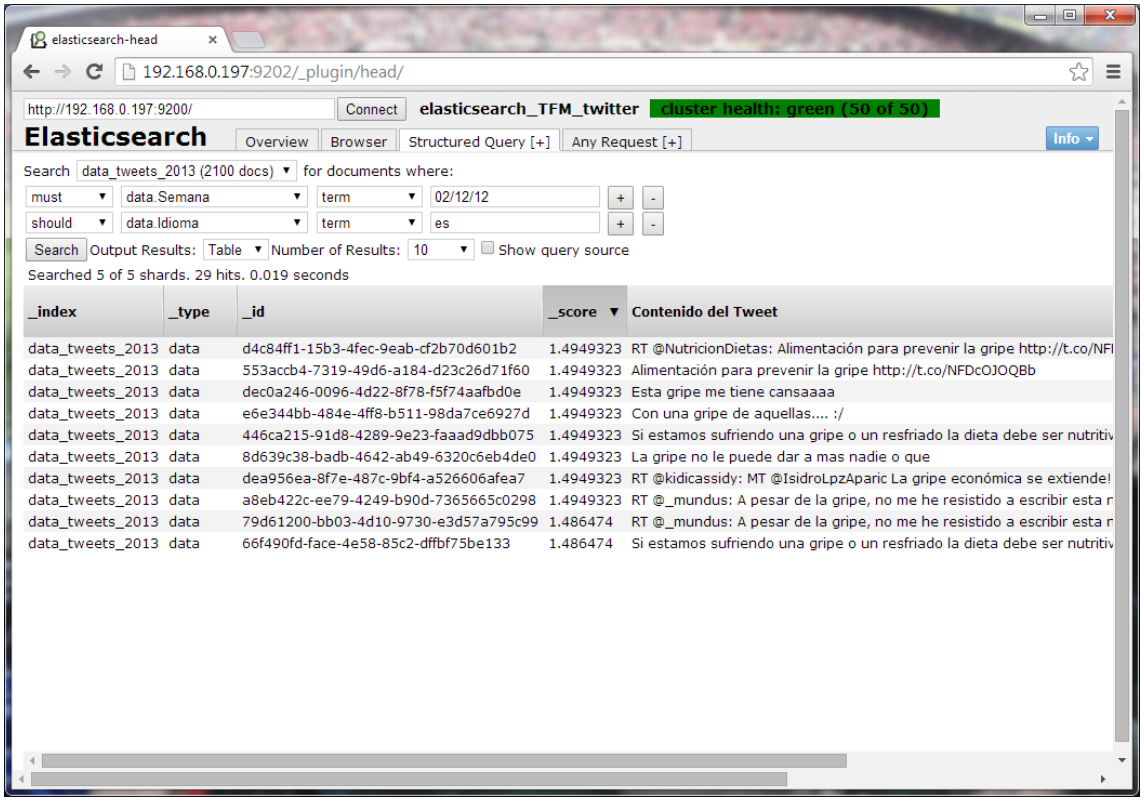

Figura 10 - Consulta plugin head

- **BIGDESK** [13]: Aquest *plugin* fa un anàlisi exhaustiu del clúster del sistema, donant especial importància a tots els aspectes que intervenen en el bon funcionament del mateix. La comanda utilitzada per tal d'instal·lar-lo en el node màster del sistema és la següent:

```
$ ./bin/plugin -install lukas-vlcek/bigdesk/2.4.0
```

Així, les primeres gràfiques, que es mostren a la figura 11, resumeixen el rendiment de la màquina virtual de Java, on es poden detectar possibles anomalies o saturacions en els nodes, les diferents cues dels fils d'execució (cerques, índexs, etcètera) i la monitorització del *hardware* de cada node participant en el clúster. Per tal de completar la informació, les següents gràfiques, il·lustrades a la figura 12, expressen la quantitat de processos, índexs i comunicacions (principalment del protocol HTTP), per acabar resumint el sistema de fitxers utilitzat, amb el volum de lectures i escriptures (variable *Delta*).

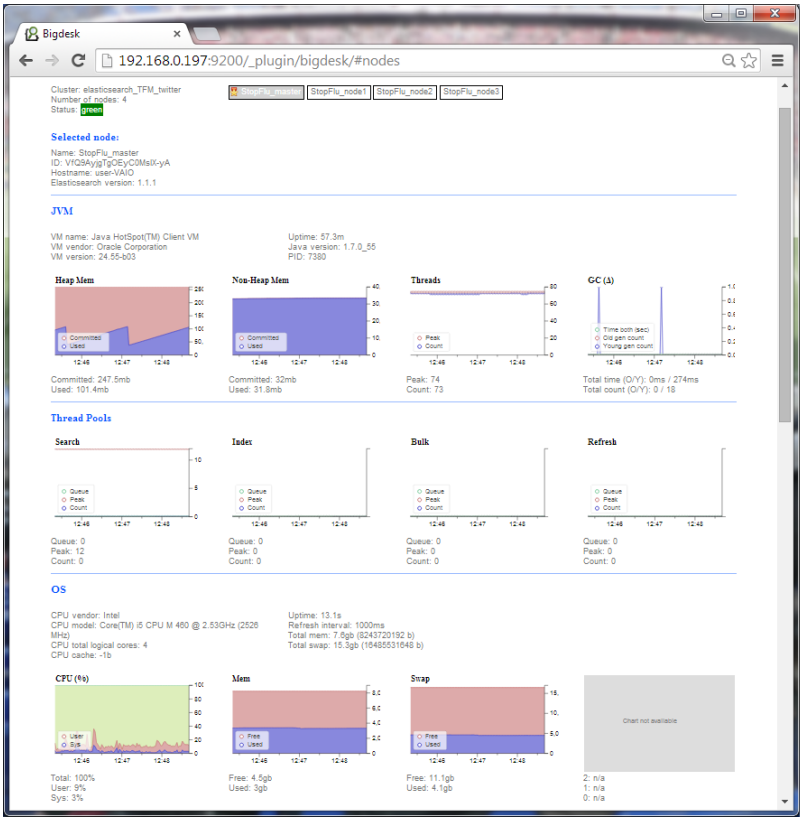

Figura 11 - BigDesk JVM, Thread Pools and OS

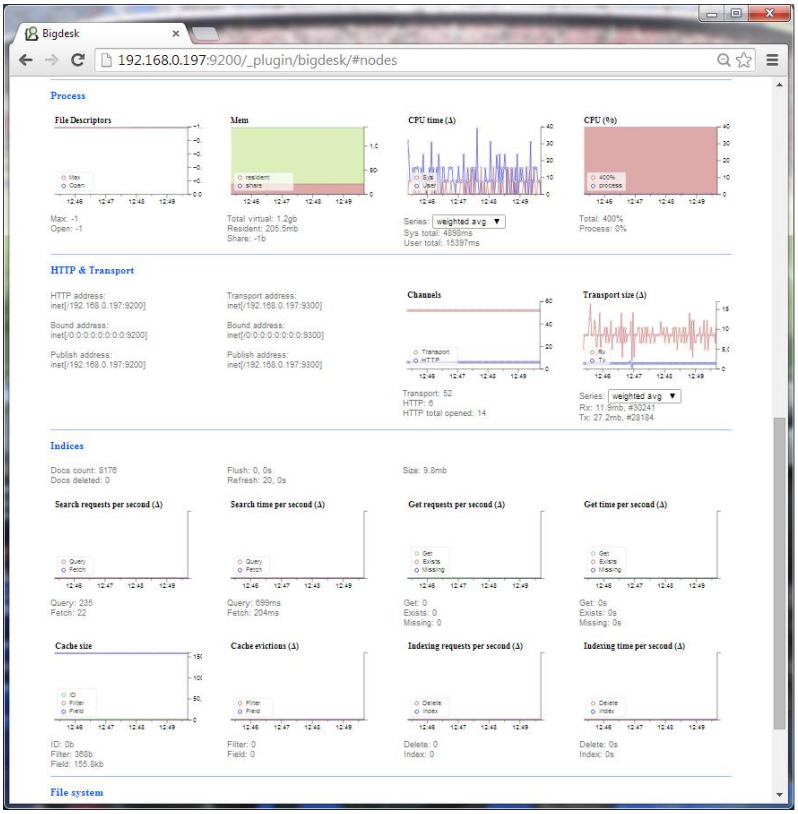

Figura 12 - BigDesk Process, HTTP and Indices

Finalment, a l'apartat '*cluster*', el *plugin* presenta un diagrama, en fase experimental, sobre la distribució de les dades dintre del sistema i, més concretament, a cadascun dels nodes que el formen. Aquesta gràfica remarca de manera clara el balanceig de càrrega i l'emmagatzematge de dades per tal de cercar una bona estabilitat del conjunt. Un dibuix desigual i heterogeni serà sinònim d'una mala gestió dels recursos disponibles per l'ús d'aquesta tecnologia. Caldrà pensar, doncs, en mesures correctores i viables a curt i llarg termini.

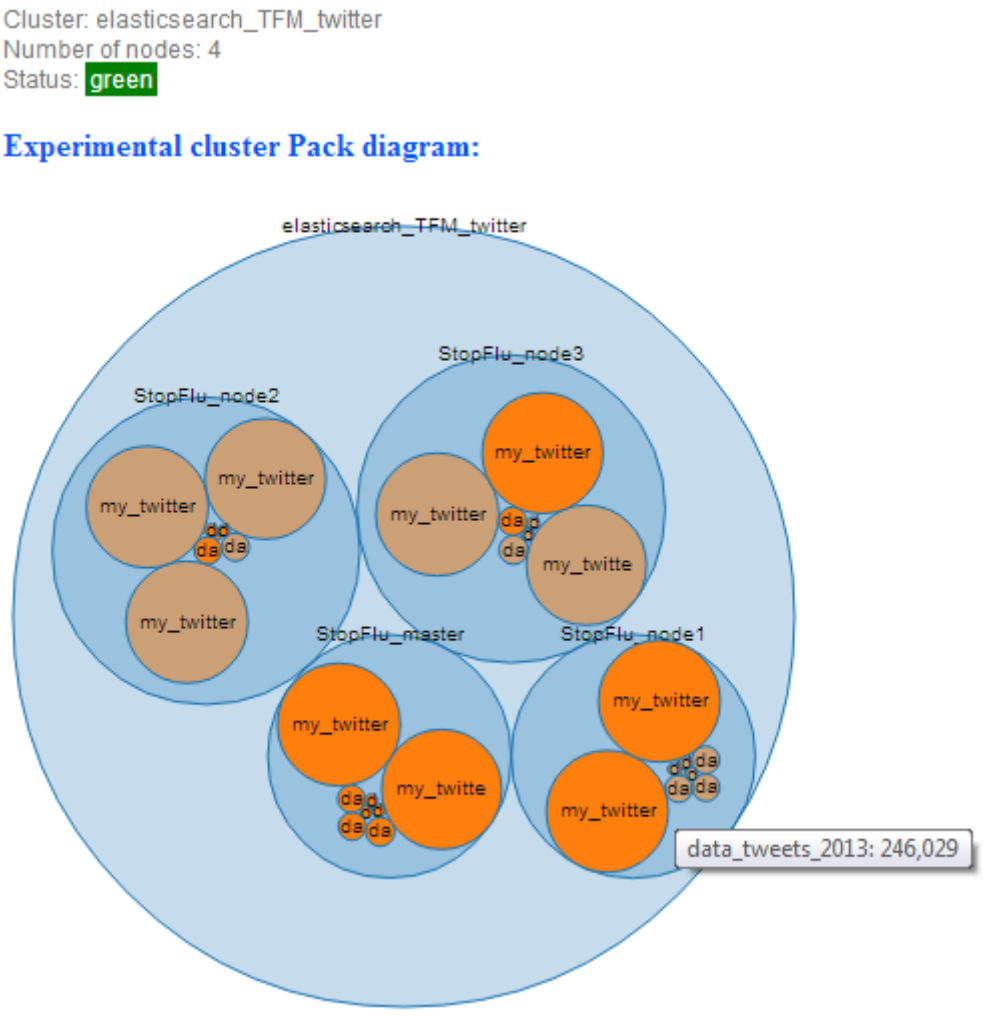

Figura 13 - BigDesk cluster diagram

- **ElasticHQ** [14]: Es tracta d'un servei de monitorització exhaustiu que recull les funcionalitats presentades en els *plugins* anteriors. Els punts forts d'aquest *plugin* es poden resumir amb el següent quadre:

Dades en temps real	Amb actualitzacions de 5 segons, es refresca la pantalla per obtenir la informació més recent.
Monitorització dels clústers	Seguiment de la salut del sistema en temps real, amb dades generals i estadístiques destriables.
Monitorització dels nodes	Seguiment dels nodes de manera individualitzada, per tal d'aplicar canvis en el rendiment E/S per l'ús de la memòria. Són el centre del sistema.
Gestió dels índexs	Actualitzar, optimitzar, eliminar i veure les mètriques dels índexs mitjançant un simple enllaç. Se'n detalla informació de fragments i àlies també.
Cerca i consulta	La funció de cerca flexible fa més fàcil el fet de buscar en un índex o índexs.
Rest API UI	Es deixen de banda comandes tipus cURL, APIs REST i formats JSON. La potent REST UI realitza la feina automàticament.
Revisió diagnòstica	Analitza mètriques clau mitjançant tots els nodes, oferint solucions a problemes comuns i consells útils.
Disseny multi-plataforma	Implementat amb el *BootStrap* de *Twitter*, ofereix una bona resposta davant els navegadors més actuals i una bona usabilitat amb la majoria de dispositius mòbils.
Sense software	Es pot executar directament al navegador, el que permet una supervisió del clúster sense necessitat d'instal·lar cap tipus de programari.

Tal com es mostra a la figura 14, la plana principal d'aquest *plugin* sintetitza de forma molt acurada tota la informació rellevant del sistema implementat i distribueix les diferents funcionalitats en àrees molt ben definides. L'apartat de diagnosi de nodes facilita la resolució de problemes sorgits en l'ús diari del sistema, i el sistema de cerca dintre dels índexs és un dels punts forts de la implementació feta, amb una potència i rapidesa en tot tipus d'entorns molt destacable. Per contra, el punt més feble del *plugin* resideix en un diagrama del clúster, basat en els diagrames d'arcs, poc entenedor i rebuscat, sobretot per sistemes amb una gran quantitat de nodes, índexs, fragments i rèpliques (Figura 15).

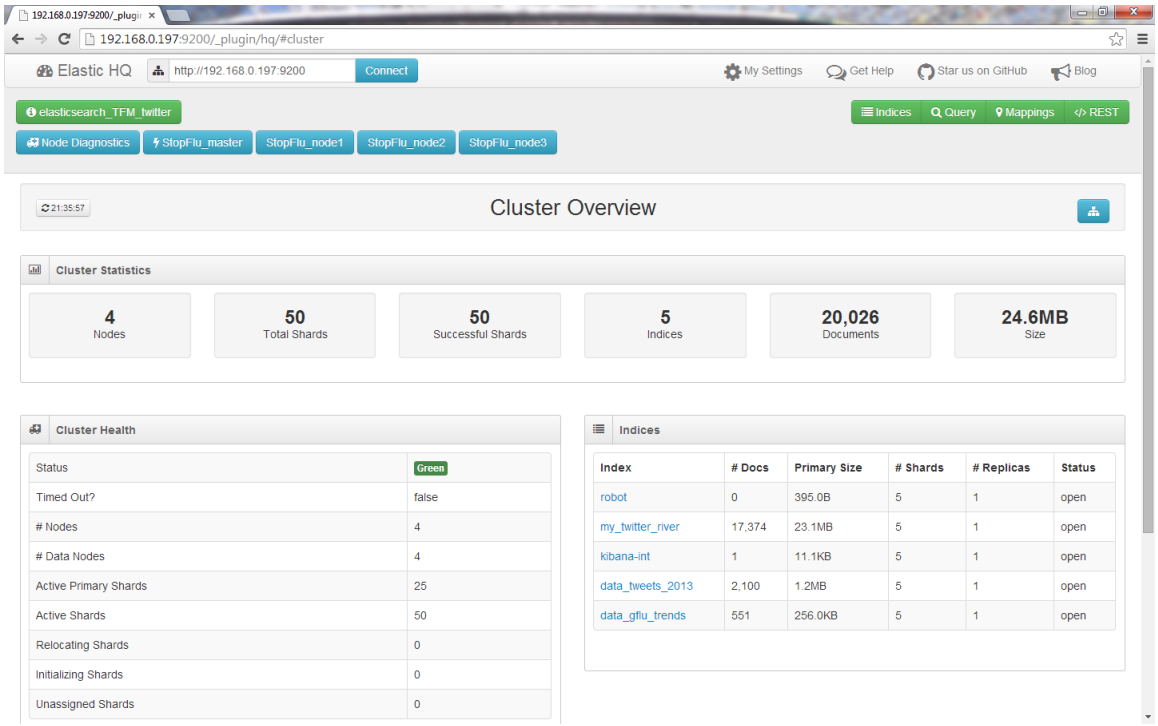

Figura 14 - ElasticHQ Cluster overview

Encara que aquest *plugin* no pertany a l'equip de *Elasticsearch*, està recolzat per una comunitat d'usuaris destacable, que introdueixen millores i solucionen els *bugs* (errors més o menys importants que sorgeixen arran d'una mala implementació del codi utilitzat) que sorgeixen en tota aplicació depenent de múltiples variables.

A la seva plana web, tenen un apartat d'estadístiques on s'aprecia l'evolució d'aquesta tecnologia dintre del sector de la gestió de grans volums de dades [15].

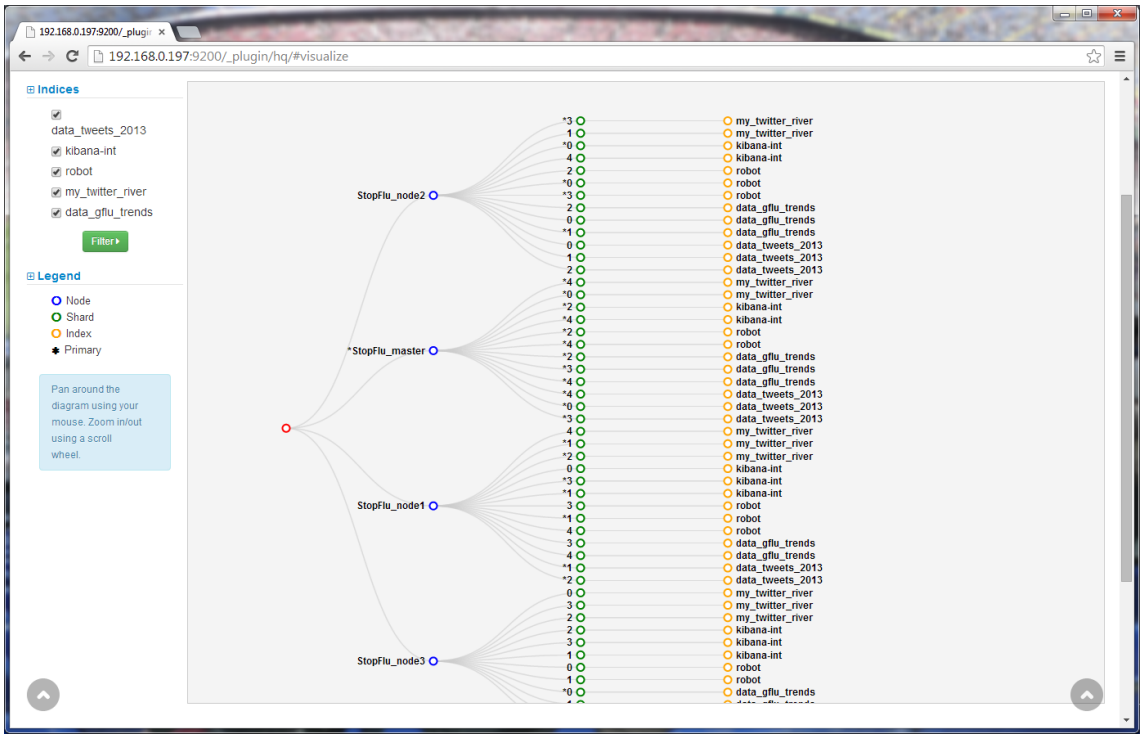

Figura 15 - ElasticHQ diagram system

Tots aquests *plugins* de gestió i monitoratge, els analitzats i algun més realitzat per la comunitat, com pot ésser *Paramedic Plugin* [16] o *Whatson Plugin* [17], estan molt treballats i arriben a un grau de funcionalitat prou satisfactori. Però l'equip de *Elasticsearch* es va veure en la necessitat de desenvolupar un producte 100% natiu, construït des de zero pels programadors que coneixen realment l'arquitectura del producte i poden facilitar una eina segura i fiable a tot tipus de client. Així, *Marvel* [18] (finals de gener del 2014) resolt un dels reptes més importants per *Elasticsearch*: com obtenir una visió completa de l'estat de la implementació i la forma de mantenir els clústers optimitzats. Encara que Elasticsearch exposa un conjunt molt ric d'estadístiques del sistema mitjançant la seva '*Stats API'*, la traducció d'aquests resultats en informació que es pugui processar, és una qüestió molt diferent. Marvel ofereix, tant en temps real com historificat, la tan necessària visibilitat del sistema implementat. D'aquesta manera, es desmarca de les solucions

presentades anteriorment i genera negoci oferint el producte mitjançant la compra de llicències per entorns de producció.

Marvel és un *plugin* per *Elasticsearch* que es connecta al nucli dels clústers i immediatament comença a enviar les estadístiques i els esdeveniments de canvi. Per defecte, aquests esdeveniments s'emmagatzemen al mateix clúster. No obstant, es poden enviar a un altre clúster si es necessita. Un cop les dades s'extreuen i s'emmagatzemen, el segon aspecte de *Marvel* entra en acció, un quadre de comandament específic confeccionat per tal de donar una visió clara de l'estat del sistema i proporcionar les eines necessàries per arribar als indrets més profunds de *Elasticsearch*.

La instal·lació d'aquest *plugin* es realitza a tots els nodes participants del clúster mitjançant la següent comanda:

```
$ ./bin/plugin -install elasticsearch/marvel/latest
```

A la plana principal de *Marvel* (figura 16) s'accedeix amb la següent adreça web:
`http://localhost:9200/_plugin/marvel`

El panell de control mostra les mètriques essencials per tal de mantenir el clúster en condicions òptimes. També proporciona una visió general dels nodes i dels índexs, que es mostren en dos apartats diferenciats, junt amb les mètriques més destacables. Aquestes taules serveixen com a punt de partida per detallar més estadístiques sobre els nodes i els índexs, on es poden observar més de 90 variables diferents traçades en temps real. Cal destacar que es poden seleccionar varis nodes/índexs per comparar resultats i traslladar-los a una nova ubicació per tal de guanyar visibilitat al conjunt.

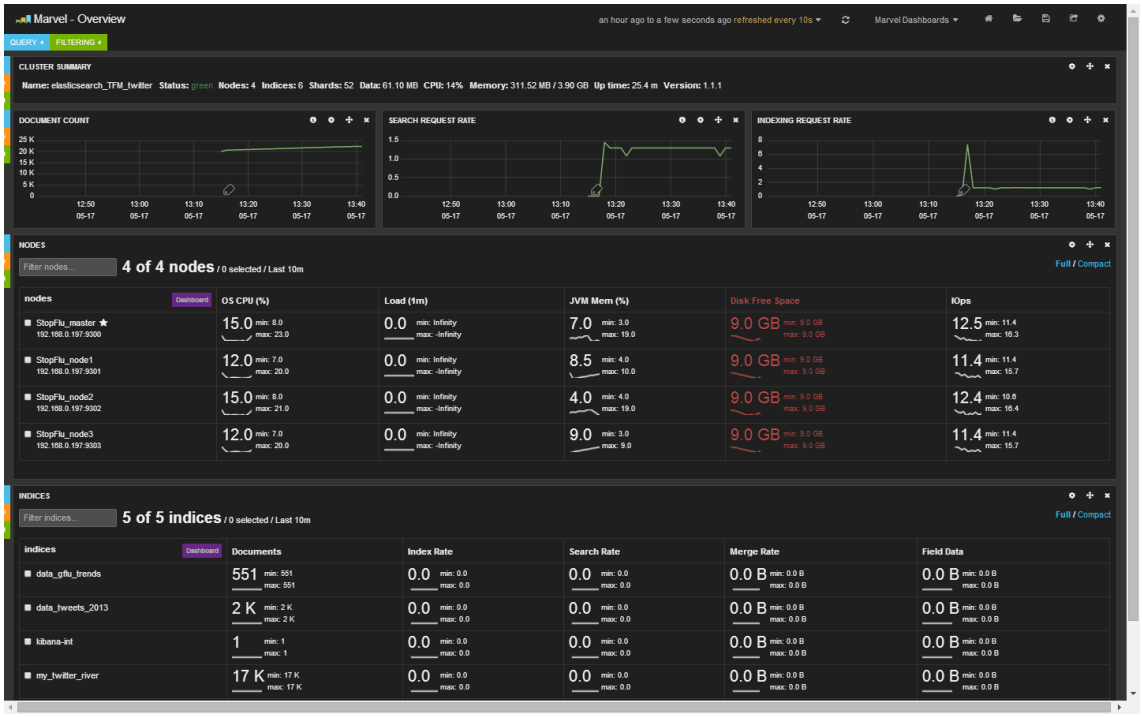

Figura 16 - Marvel Overview

La plana d'estadístiques d'un node mostra les gràfiques específiques del mateix. Aquestes inclouen mètriques de nivell de *hardware* (com la càrrega i l'ús de CPU), els processos i les estadístiques de JVM (ús de la memòria, GC), i mètriques més concretes com l'ús de dades, peticions de cerca i la taxa de rebuig del grup de subprocessos (Figura 17).

La plana d'estadístiques d'índexs és molt semblant a la dels nodes. Les mètriques són per índex, amb dades agregades de tots els nodes del clúster. Per exemple, la gràfica de la mida d'emmagatzematge mostra la mida total de les dades d'un índex a través de tot el clúster (Figura 18).

Figura 17 - Marvel node stats

Figura 18 - Marvel index stats

El panell d'esdeveniments del clúster mostra qualsevol esdeveniment de cert interès que succeeix al sistema. Els esdeveniments típics poden ser des de

inclusió i eliminació de nodes al clúster, fins a la creació d'un índex nou. Es considera la finestra d'entrada al sistema nerviós de *Elasticsearch*.

Marvel també ve amb una consola de desenvolupament lleuger, basat en la popular extensió de *Chrome* anomenada *Sense* [19]. Aquesta consola és molt útil a l'hora de realitzar una crida addicional a la API per tal de comprovar quelcom o modificar un ajust. Aquesta entén tant JSON com la API pròpia.

Finalment, un riu (*river*) és un servei connectable que funciona dintre d'un clúster per l'extracció de dades que s'indexen a mesura que són recollides. Es composa d'un nom únic i d'un tipus. El tipus és el tipus del riu (out *of the box, dummy*) i el nom identifica de forma exclusiva el riu dintre de l'agrupació. Una instància de riu (i el seu nom) és un tipus dintre de l'índex '*_river'*. Tots accepten a les seves implementacions un document anomenat '*_meta*' que l'associa amb el tipus de riu que té (ja sigui de *twitter*, *couchdb* [20], *wikipedia*, etcètera). La creació d'un riu es fa mitjançant una senzilla petició de tipus *curl* que indexa el document meta. El rius són únics dintre de l'agrupació. Ells s'assignen automàticament a un dels nodes i s'executen. Si aquest falla, s'assignarà automàticament a un altre node.

En el següent apartat es planteja el cas concret del riu de *twitter*, que havia de ser el subministrador oficial de dades del projecte.

3.4 Exprimint el *Twitter*

La font principal d'adquisició de dades pel sistema BI plantejat havia de ser el *Twitter* en un primer moment. Gràcies al impacte que tenen avui en dia les xarxes socials en les vides d'un gran percentatge de la població mundial, es podien extreure tots els comentaris referents a uns criteris determinats i, així, realitzar un estudi en funció dels missatges publicats pels usuaris d'aquesta plataforma i la informació obtinguda per un centre hospitalari local. Seguint amb l'anàlisi de *Elasticsearch* i les seves funcionalitats, s'analitza la utilitat del *river* de *Twitter* [21] en funció de les necessitats del projecte.

Es realitza una primera extracció de dades observant que aquestes només indexen els missatgers actuals via *streaming* [22] (mitjançant l'API pública de *Twitter*). Per establir la connexió amb la API de *Twitter*, cal registrar-se com a usuari de la plataforma i accedir a la part de desenvolupadors [23]. Aquí es poden demanar les claus privada i pública per tal d'interactuar entre l'aplicació pròpia i la API pública. Un cop creada una aplicació de prova, les dades generades són les següents:

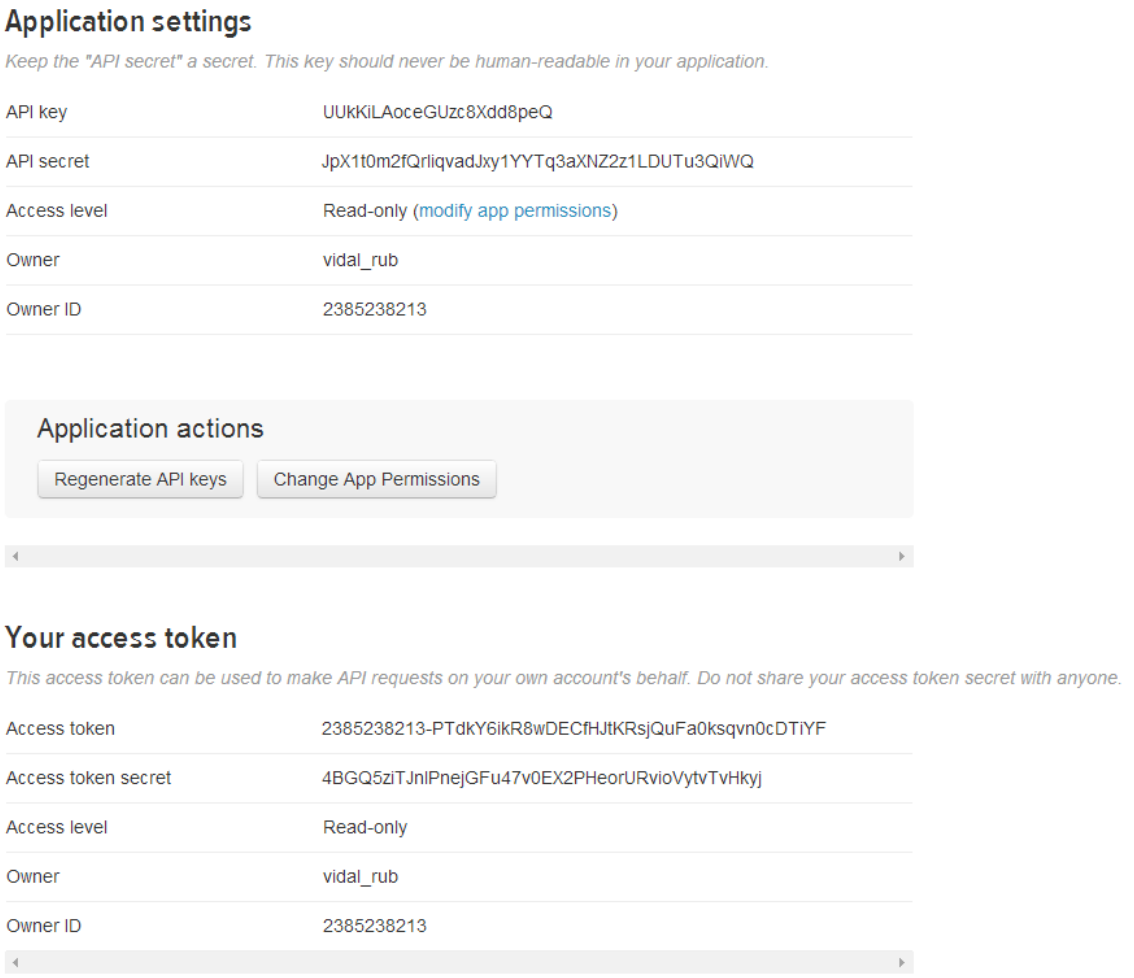

Figura 19 - Dades d'usuari API Twitter

Amb aquesta sèrie de dades ja es pot configurar el *script* que activa el riu i connecta el sistema BI amb la API de *Twitter* per la recollida de missatges. Aquest fa una petició creant un índex de tipus '*twitter*' que guardarà en diferents camps tots els missatges que es vagin generant des de la creació del riu fins a què aturem el mateix amb la comanda:

```
curl -XDELETE http://localhost:9200/_river/my_twitter_river/
```

El codi utilitzat per la creació del riu és el següent:

```
curl -XPUT localhost:9200/_river/my_twitter_river/_meta -d '
{
    "type" : "twitter",
    "twitter" : {
        "oauth" : {
            "consumer_key" : "UUkKiLAoceGUzc8Xdd8peQ",
            "consumer_secret" : "JpX1t0m2fQrliqvadJxy1YYTq3aXNZ2z1LDUTu3QiWQ",
            "access_token" : "2385238213-PTdkY6ikR8wDECfHJtKRsjQuFa0ksqvn0cDTiYF",
            "access_token_secret" : "4BGQ5ziTJnlPnejGFu47v0EX2PHeorURvioVytvTvHkyj"
        }
    },
    "index" : {
        "index" : "my_twitter_river",
        "type" : "status",
        "bulk_size" : 100,
        "flush_interval" : "5s"
    }
}
'
```

El principal problema que presenta aquest riu és el fet de no poder buscar missatges anteriors a la data actual, ja que *Twitter* va decidir darrerament vendre el seu històric a empreses externes per a la seva explotació; així doncs, aquesta funcionalitat no serveix pels propòsits d'aquest projecte. S'ha valorat la utilització d'una d'aquestes empreses [24] per tal d'aconseguir la informació necessària dels anys a estudiar, però el preu del servei i la complexitat d'implementació de la API pròpia d'aquesta empresa han desaconsellat aquesta opció.

En la cerca de dades contrastades per l'anàlisi, s'ha seleccionat el registre publicat al projecte de *Google* [25] sobre l'evolució del virus de la grip a tot el món, un recull de cerques de missatges de *Twitter* utilitzant el cercador avançat [26] de la mateixa empresa i les dades proporcionades pel consultor de l'assignatura referents a un hospital de Catalunya.

Un cop es coneix l'origen de la informació, només falta implementar el sistema que actuarà de *Data Warehouse*, on guardar totes les dades exposades amb anterioritat.

3.5 Implementació del clúster

El sistema escollit consta del producte estudiat a l'apartat 3.2 (*Elasticsearch* en la versió 1.1.1) implementat en una topologia de 4 nodes executats en una mateixa màquina estàndard d'ús personal (CPU *Intel* I5 amb 8 GB de memòria RAM). Es configuren els nodes mitjançant el arxiu '*elasticsearch.yml*' (Annex 1, apartat 1), modificant les següents característiques:

Variable	**Configuració**
Clúster.name	Elasticsearch_TFM_twitter Tots els nodes han de tenir el mateix nom de clúster per tal d'establir una comunicació entre ells.
Node.name	StopFlu_master, StopFlu_node1, StopFlu_node2 i StopFlu_node3, respectivament.
Node.master	El primer node agafa el valor de '*true'* i els demés prenen el valor de '*false'.* Així comença el clúster, ja que necessita un node màster d'inici. Si aquest caigués, s'escolliria un altre automàticament.

Node.data	Tots els nodes tenen el valor a '*true'*. Es podria utilitzar el node màster només com a coordinador, si el sistema fos més complex.
Transport.tcp.port	S'assignen els ports 9300, 9301, 9302 i 9303, respectivament.
http.port	S'assignen els ports 9200, 9201, 9202 i 9203, respectivament.
Index.number_of_replicas	2, per estudiar el comportament de les mateixes en diferents etapes del clúster (addició, caiguda de nodes, etcètera).

Existeixen molts més paràmetres de configuració pels nodes, però s'ha establert aquesta configuració per tal de tenir un sistema estable amb els recursos disponibles. Així s'han agafat el nombre de fragments (5) i la ruta d'emmagatzematge de dades amb els valors per defecte.

La parametrització de la màquina virtual de Java (JVM), amb la corresponent revisió dels *logs*, pot ajudar a optimitzar el clúster i ajustar-lo a la maquinària utilitzada en cada entorn de treball. També les opcions de '*discovery*' permeten configurar els nodes per tal de localitzar-los en infraestructures de tercers (EC2 de *Amazon* [27] i GCE de *Google* [28], entre d'altres). Queda palès, doncs, que es tracta d'una tecnologia d'avantguarda, modular, escalable i altament distribuïble.

La inicialització del clúster es realitza mitjançant un *script* força senzill que arrenca cadascun dels nodes, anomenat 'start_elasticsearch_cluster.bat':

```
start call "C:/BI_StopFlu/elasticsearch-1.1.1/bin/elasticsearch.bat"

PING -n 1 -w 15000 1.1.1.1>NUL

start call "C:/BI_StopFlu/elasticsearch-node1/bin/elasticsearch.bat"

PING -n 1 -w 15000 1.1.1.1>NUL

start call "C:/BI_StopFlu/elasticsearch-node2/bin/elasticsearch.bat"

PING -n 1 -w 15000 1.1.1.1>NUL

start call "C:/BI_StopFlu/elasticsearch-node3/bin/elasticsearch.bat"
```

Amb aquesta sèrie de comandes, aconseguim obrir quatre consoles de MS-DOS, on cadascuna arrencarà un node del sistema, amb un interval de temps establert en 15 segons per tal que el màster reconegui els nodes restants a mesura que s'inicialitzen. Aquest interval de temps també permet que els diferents nodes carreguin els *plugins* associats indispensables pel nostre sistema.

3.6 Càrrega de dades al sistema

Per tal de carregar tota la informació recopilada, s'utilitza un riu (*river*), explicat en l'apartat 3.3, que permet indexar fitxers en format CSV en el sistema *Elasticsearch*. La instal·lació del riu es realitza mitjançant la comanda:

```
bin/plugin -install river-csv –url
https://github.com/AgileWorksOrg/elasticsearch-river-
csv/releases/download/2.0.1/elasticsearch-river-csv-2.0.1.zip
```

Un cop el *plugin* està instal·lat als diferents nodes que conformen el sistema, qualsevol d'ells pot escoltar la petició d'execució del riu i començar a carregar dades en el repositori que té assignat. Tot seguit, de forma automàtica, el sistema realitza els ajustaments necessaris per tal d'assegurar fragments i rèpliques.

El codi utilitzat per crear el riu és:

```
curl -XPUT localhost:9200/_river/my_csv_river/_meta -d '
{
    \"type\" : \"csv\",
    \"csv_file\" : {
        \"folder\" : \"/tmp\",
        \"first_line_is_header\" : \"true\"
                    \"field_separator\" : \";\"
                    },
         \"index\" : {
                    \"index\" : \"data_tweets_2013\",
                    \"type\" : \"data\"
                    }
}'
```

Un cop tenim les dades carregades, s'ha de tancar el riu amb la comanda següent:

```
curl -XDELETE http://localhost:9200/_river/my_csv_river/
```

Arribats a aquest punt, el sistema BI ja disposa d'un *Data Warehouse* estable i amb les dades necessàries per realitzar l'estudi plantejat en l'inici del projecte. La figura 20 il·lustra aquest fet, on podent distingir els diferents índexs creats. En especial, interessen el 'data_hospitalitzacions' i el 'data_tweets_2013', que emmagatzemen les dades analitzables mitjançant l'explotació de les dades. Els dos primers índexs, els de més a la dreta, són creats per *Marvel* per tal de crear el registre històric de monitoratge del sistema *Elasticsearch*.

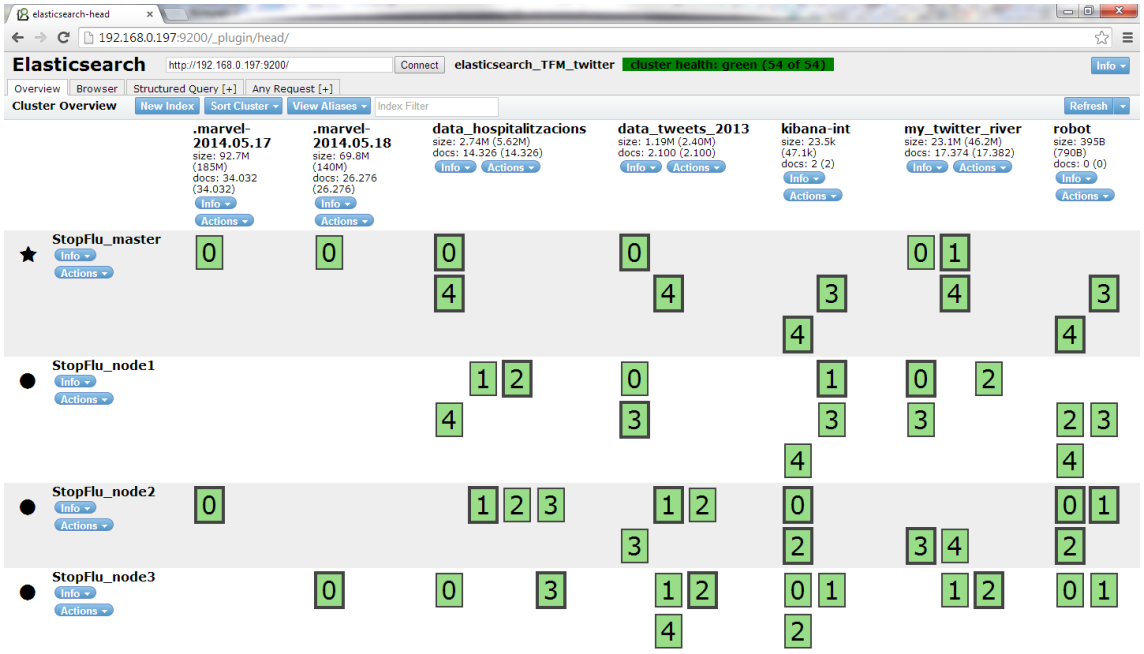

Figura 20 - Data Warehouse del sistema BI

3.7 Quadre de comandament

Cercant la total compatibilitat de les diferents eines utilitzades, el quadre de comandament, on es donarà visibilitat als resultats obtinguts, s'implementa en un producte anomenat *Kibana*. És la interfície gràfica natural de tot sistema *Elasticsearch* (el producte *Marvel* empra aquesta tecnologia - figures 16,17 i 18). Està escrit completament en llenguatge HTML i *JavaScript*, necessitant només un servidor web normal i destacant per la seva flexibilitat i potència.

La posada en marxa d'aquesta interfície es realitza de la següent manera:

1. Descàrrega de la última versió de *Kibana* [29] (versió 3.1.0 – maig del 2014).
2. Configuració de l'arxiu '*config.js*' (Annex 1, apartat 2), amb la modificació de la IP del sistema.
3. Còpia del contingut del directori extret al servidor web.

Cal apuntar que aquest producte enllaça directament amb el sistema *Elasticsearch* des del navegador. Per tant, aquest últim interactua directament amb el sistema, sense la intervenció de cap intermediari. És possible haver de configurar un *proxy* invers per tal de restringir l'accés a *Elasticsearch*.

La corba d'aprenentatge d'aquesta plataforma creix exponencialment en un temps força curt, ja que la distribució dels diferents elements bàsics està molt optimitzada. Un dels punts més interessants del producte és la modularitat i flexibilitat a l'hora de presentar els resultats. Per mitjà dels apartats 'consultes' i 'filtres', es pot arribar a confeccionar una depuració de les dades seleccionades pel posterior anàlisi. Així, *Kibana* s'organitza en un sistema de files i panells. Actualment existeixen 11 tipus de panells per tal de cobrir el ventall d'opcions que poden sorgir a l'hora de dissenyar la visualització dels resultats d'una o vàries consultes. Aquests es poden afegir, eliminar i reorganitzar dintre de la plana segons les necessitats de l'usuari o el desenvolupador. A més, inclou l'opció de bloquejar l'estructura plantejada o deixar-la oberta per futures modificacions.

Quan el quadre de comandament està acabat, es pot guardar mitjançant el botó corresponent situat a la cantonada superior dreta. D'aquesta manera, es poden tenir diferents escenaris que aporten versatilitat en representacions i discussions dels resultats. També s'ofereix la possibilitat d'exportar esquemes de quadres en format JSON i utilitzar-los en altres sistemes similars i, fins i tot, realitzar un quadre de comandament propi des de zero confeccionant tots els panells, files i configuracions mitjançant la programació de plantilles i *scripts* en format JSON.

4. Explotació de les dades

L'últim punt dels objectius del treball s'enfoca en l'explotació de les dades mitjançant el quadre de comandament plantejat amb el sistema *Elasticsearch + Kibana*. Com es pot observar a la figura 21, el quadre de comandament realitzat cerca una connexió entre el nombre de missatges publicats i les hospitalitzacions efectuades durant la temporada hivernal de 2012-2013. En aquest sentit, el panell 'histograma' es presenta com a principal dada gràfica perquè recull de manera molt intuïtiva la tendència dels dos tipus de dades (de color groc les hospitalitzacions i de color verd els *tweets*).

Els KPI emprats per les sèries de dades són:

- Hospitalitzacions: el codi GRD, amb valor 541, que agrupa a pacients ingressats per una malaltia respiratòria, excepte certes infeccions o inflamacions pulmonars, bronquitis aguda o asma, i que, a més, tenen un altre diagnòstic etiquetat de complicació o comorbiditat major com insuficiència respiratòria aguda, pneumònia o insuficiència renal aguda.

- Twitter: En el cercador avançat de la plataforma, es va utilitzar la paraula 'gripe' com a criteri principal en les diferents consultes mensuals.

Un dels punts forts d'aquest sistema és la gran velocitat aconseguida en les diferents interaccions realitzades sobre el quadre. Com a exemple, es pot marcar sobre l'histograma principal la franja del mes de febrer de l'any 2013 i, quasi de forma instantània, s'actualitzen tots els panells que tenen informació relacionada amb aquest canvi de paràmetre. Per últim, el panell '*stable*' permet tenir un control sobre tots els registres que compleixen els requisits imposats per les cerques i filtres sobre els diferents índexs que s'executen al quadre. D'aquesta manera, no es fa necessària la utilització dels *plugins* de *Elasticsearch* un cop el sistema està ben definit (Figura 22).

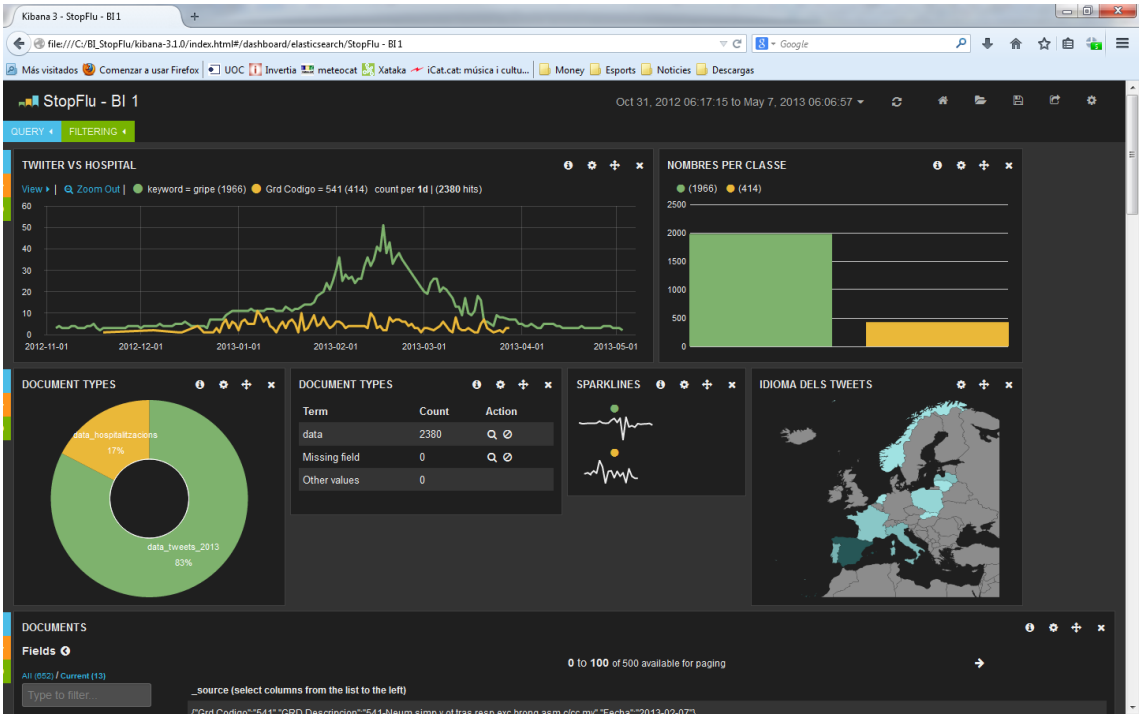

Figura 21 - StopFlu Dashboard I

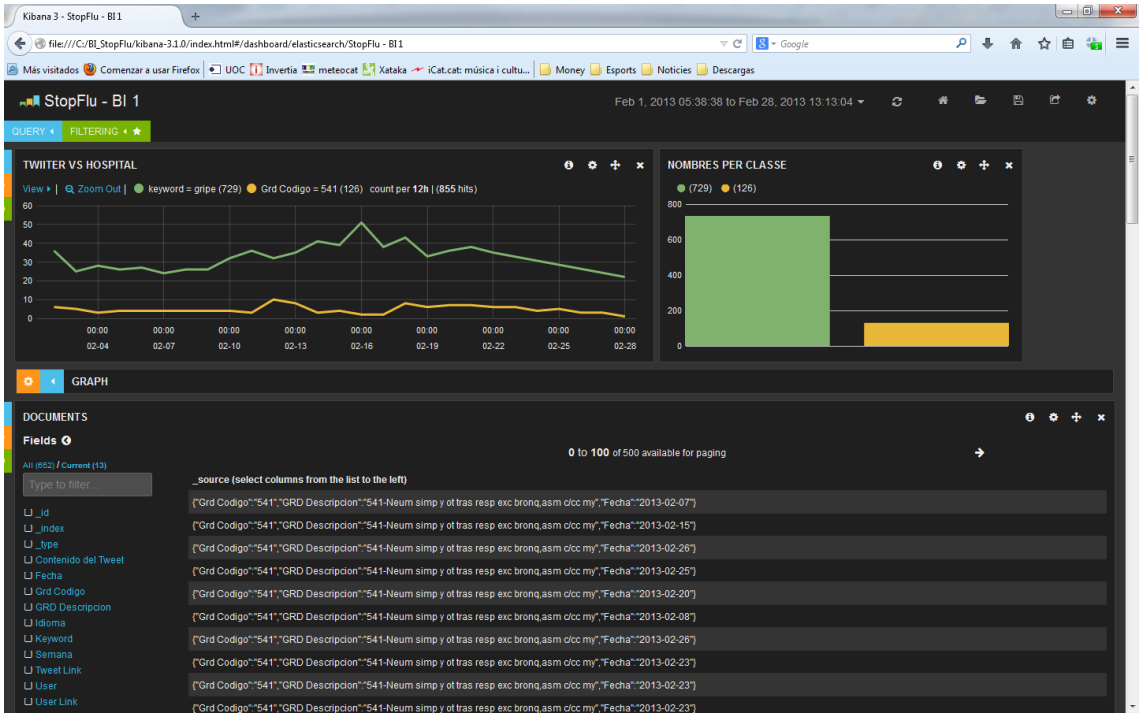

Figura 22 - Detall mes de febrer 2013

5. Conclusions

En aquest treball, s'ha volgut investigar una nova manera de construir un sistema BI sortint del circuit convencional. Les ganes d'aprofundir en productes d'avantguarda i cercar solucions 100% parametritzables, m'han portat a descartar els productes més coneguts i comercials de *Business Intelligence*, i escollir una solució no tan coneguda però que es podia ajustar als objectius inicials (recollida d'informació, càrrega de dades, creació d'un quadre de comandament i la posterior explotació de les dades). Un dels punts que més m'atreia d'aquest projecte era el fet de poder gestionar grans volums de dades i controlar-les en una estructura estable, modular, distribuïda i escalable. Tots aquests elements em van portar a escollir la plataforma treballada i m'han ajudat a consolidar molts dels coneixements adquirits al màster, així com a experimentar amb solucions que poden ésser útils en l'àmbit laboral actual i futur.

El plantejament del treball ha sigut arriscat des del principi i això m'ha portat a llegir molta documentació i anar solucionant problemes a mesura que m'anaven sorgint. L'aprenentatge funcional de tot el sistema m'ha ocupat més temps del desitjat i això m'ha impedit aprofundir, com m'hagués agradat, en l'anàlisi i explotació de les dades, que reconec, és la base d'un sistema BI. Considero que el programari emprat pot realitzar de manera brillant el cas estudiat i, possiblement, els resultats no són tan senzills d'aconseguir com en un principi em pensava. S'ha de dir que la plataforma ELK està pensada per treballar fonamentalment amb registres recollits per plataformes concretes (*Logstash*, *Fluentd*, etcètera), que compleixen uns estàndards concrets que no tenien les fonts d'informació d'aquest projecte. La restricció imposada per *Twitter* sobre la consulta de les seves dades històriques també ha suposat un replantejament del model inicial i m'ha consumit un temps preciós no planificat prèviament.

Pel que fa a la planificació i metodologia de treball, m'he ajustat en la mesura del possible als terminis reflectits al pla de treball. Aquest últim es va

entregar amb uns dies de retard per culpa de diferents compromisos laborals, i he hagut de realitzar un esforç suplementari per anar recuperant el ritme de treball, per tal de posar-me al dia respecte a la planificació inicial. Com ja he comentat anteriorment, el punt de l'explotació i anàlisi de dades es queda en un estadi inicial. En línies de treball futur, es podria aprofundir en la rectificació del sistema actual i la confecció d'altres quadres de comandaments, amb aquestes i d'altres dades, amb l'ajuda de les pautes explicades en aquest treball. Per altra banda, m'hagués agradat provar el sistema en un entorn descentralitzat, tot disposant de nodes al núvol (comentat a l'apartat 3.5), per tal de comprovar l'escalat i la rapidesa de processament promesa pels desenvolupadors del producte. La durada d'aquest projecte (poc més de 3 mesos) tampoc ajuda a plantejar un tests més robustos (càrrega massiva de dades, augment de la complexitat de la topologia del sistema, etcètera) que certifiquin l'adequació d'aquesta solució en entorns de producció crítics. Amb un equip suficient i un projecte ben estructurat, es poden arribar a desenvolupar sistemes predictius, i de BI en general, personalitzats per cada funció que requereixi qualsevol àrea d'una empresa.

6. Glossari

- **Apache**: Comunitat descentralitzada de desenvolupadors, creada a l'any 1999, que treballen en els seus propis projectes de *software* de codi obert. Els projectes es caracteritzen per un model de desenvolupament basat en el consens i la col·laboració, dintre d'una llicència oberta i pragmàtica.
- **API**: Interfície de programació d'aplicacions, és el conjunt de funcions i procediments (o mètodes, a la programació orientada a objectes) que ofereix una certa biblioteca per ésser utilitzada per un altre software com a una capa d'abstracció.
- **BI**: Acrònim del concepte '*Business Intelligence'*, que descriu a la intel·ligència de negoci com el conjunt d'estratègies i aspectes rellevants enfocats a l'administració i creació de coneixement sobre el medi, a través de l'anàlisi de les dades existents en una organització o empresa.
- ***Big Data***: És, en el sector de les TIC (tecnologies de la informació i la comunicació), una referència als sistemes que manipulen grans volums de dades. Les dificultats més habituals en aquests casos es centren en la captura, l'emmagatzematge, la cerca, la compartició, l'anàlisi i la visualització de les mateixes.
- **CPU**: Unitat Central de processament, és el component principal de l'ordinador i d'altres dispositius programables, que interpreta les instruccions contingudes als programes i processa les dades.
- **CSV**: Acrònim de '*Comma-Separated Values'*, són un tipus de documents en format obert que serveixen per representar dades en forma de taula, on les columnes es separen per comes i les files per salts de línia.
- ***Data Warehouse***: Magatzem de dades, és una col·lecció de dades orientada a un àmbit determinat (empresa, organització, etcètera), integrada, no volàtil i variable en el temps, que ajuda a la presa de decisions en l'entitat en què s'utilitza.

- **Elasticsearch**: Motor de cerca i anàlisi en temps real, potent, distribuït i de codi obert. Dissenyat des del principi per un ús en entorns distribuïts on la fiabilitat i l'escalat són imprescindibles.
- **E/S**: Entrada i sortida, és la comunicació entre un sistema de processament d'informació, com un ordinador, i el món exterior, possiblement un ésser humà o un altre sistema de processament d'informació. Els dispositius de E/S són utilitzats per una persona per comunicar-se amb un ordinador.
- ***Framework***: Defineix, en termes generals, a un conjunt estandarditzat de conceptes, pràctiques i criteris per enfocar un tipus de problemàtica particular que serveix com a referència, per enfrontar i resoldre nous problemes de característiques similars.
- **GC**: En Java el problema de les fugues de memòria s'evita en gran mesura gràcies a la recol·lecció de brossa. El programador determina quan es creen els objectes i l'entorn en temps d'execució de Java és el responsable de gestionar el cicle de vida dels objectes. Quan no queden referències a un objecte, aquest s'esborra i s'allibera la memòria que ocupava.
- **GRD**: Grups relacionats pel diagnòstic, són una eina de gestió normalitzada, en la que mitjançant un programa informàtic, alimentat amb les dades dels pacients donats d'alta hospitalària, podem classificar als mateixos en grups clínicament similars i amb un consum de recursos sanitaris similars.
- ***GitHub***: És un sistema de control col·laboratiu de revisió i desenvolupament de software.
- ***Hardware***: Es refereix a totes les parts tangibles d'un sistema informàtic; els seus components són elèctrics, electrònics, electromecànics i mecànics. Són cables, cabines, perifèrics de tot tipus i qualsevol altre element físic involucrat.
- **HTML**: sigles de *HyperText Markup Language*, fa referència al llenguatge de marcat per l'elaboració de pàgines web.
- **HTTP**: És el protocol utilitzat en cada transacció de la *Word Wide Web*.

- **Intel**: És el major fabricant de circuits integrats del món i el creador de la sèrie de processadors 'x86', els més comuns a la majoria de les computadores personals existents.
- **JAVA**: És un llenguatge de programació de propòsit general, concurrent, orientat a objectes i basat en classes que fou dissenyat específicament per tenir tan poques dependències d'implementació com fos possible.
- **JVM**: Màquina virtual de Java, és una màquina virtual de procés natiu, és a dir, executable en una plataforma específica, capaç d'interpretar i executar instruccions expressades en un codi binari especial, el qual es generat pel compilador del llenguatge Java.
- **JSON**: Acrònim de *JavaScript Object Notation*, és un format lleuger per l'intercanvi de dades. És un subconjunt de la notació literal d'objectes de *JavaScript* que no requereix l'ús de XML.
- ***Kibana***: És el motor de visualització de dades de *Elasticsearch*, que li permet interactuar de forma nativa amb totes les seves dades mitjançant quadres de comandament personalitzats.
- **KPI**: Indicadors clau del desenvolupament, mesuren el nivell d'exercici d'un procés, indicant el rendiment dels processos, de manera que es pot assolir l'objectiu fixat.
- ***Log***: És un registre oficial d'esdeveniments durant un rang de temps concret.
- ***Marvel***: Sistema de monitoratge del sistema ELK propietari.
- ***Microblogging***: És un servei que permet als seus usuaris enviar i publicar missatges breus, generalment només de text.
- **MS-DOS**: És un sistema operatiu per computadors basats en 'x86'. Va ser el membre més popular de la família de sistemes operatius DOS de la companyia Microsoft.
- ***Open source***: És l'expressió amb que es coneix al *software* distribuït i desenvolupat lliurement.
- ***Petabyte***: És una unitat d'emmagatzematge d'informació que equival a 1.024 *Terabytes*.
- ***Plugin***: És una aplicació que es relaciona amb una altra afegint-li una funció nova i, generalment, molt específica.

- **RAM**: Acrònim de *Random-Access Memory*, s'utilitza com a memòria de treball pel sistema operatiu, els programes i la majoria de software.
- ***Restful***: És una tècnica d'arquitectura de software per sistemes hipermèdia distribuïts com la *World Wide Web*.
- ***Script***: És un programa usualment simple, que s'emmagatzema en un arxiu de text pla. Són quasi sempre interpretats i l'ús habitual és realitzar diferents tasques, com combinar components, interactuar amb el sistema operatiu o amb l'usuari.
- ***Software***: És l'equipament o suport lògic d'un sistema informàtic, que compren el conjunt dels components lògics necessaris per a fer possible la realització de tasques específiques.
- ***Streaming***: És la distribució de multimèdia a través d'una xarxa de computadors de manera que l'usuari consumeix el producte en paral·lel mentre es descarrega. Aquest tipus de tecnologia funciona mitjançant un *buffer* de dades que va emmagatzemant les mateixes a l'estació de l'usuari.
- ***Twitter***: És un servei de *microblogging*, amb seu central a San Francisco, que ha anat guanyant popularitat fins arribar a més de 500 milions d'usuaris. Permet enviar missatges de text, amb un màxim de 140 caràcters, que es mostren a la pàgina principal de l'usuari.

7. Bibliografia

Llibres:

- Jordi Conesa Caralt i Josep Curto Díaz, “Introducción al Business Intelligence”, Editorial UOC, Maig del 2010.
- Viktor Mayer-Schönberger i Kenneth Cukier, “Big Data – La revolución de los datos masivos”, Turner Noema.
  http://www.turnerlibros.com/Ent/Products/ProductDetail.aspx?ID=481
  http://big-data-book.com/
- Eric Siegel, “Analítica predictiva – Predecir el futuro utilizando Big Data”, Ediciones Anaya Multimedia, 2013.
- Phil Simon*, “Too Big to Ignore: The Business Case for Big Data”*, Wiley, April 2013.
- Judith Hurwitz, Alan Nugent, Fern Halper, Marcia Kaufman; “*Big Data For Dummies”*, Wiley, April 2013.
- Nassim Nicholas Taleb, “El cisne negro”, Editorial Planeta, 2012.

Articles:

- Jiwei Li i Claire Cardie; “*Early Stage Influenza Detection from Twitter”*, arxiv.org, Cornell University Library, Last revised 18 November 20123 (v3).
- Harshavardhan Achrekar, Avinash Gandhe, Ross Lazarus, Ssu-Hsin Yu i Benyuan Liu; “*Predicting Flu Trends using Twitter Data”*, Published in Computer Communications Workshops (INFOCOM WKSHPS), April 2011, IEEEXplore Digital Library.
- A. Sitaram i B. A. Huberman, “*Predicting the future with social media”*, *in Social Computing HP Lab*, Palo Alto, USA, 2010.

Referències Web:

[1] www.elasticsearch.org

[2] www.compass-project.org

[3] www.json.org

[4] http://www.elasticsearch.org/overview/kibana/

[5] http://lucene.apache.org/

[6] http://en.wikipedia.org/wiki/Optimistic_concurrency_control

[7] http://es.wikipedia.org/wiki/Representational_State_Transfer

[8] http://www.apache.org/licenses/LICENSE-2.0.html

[9] http://en.wikipedia.org/wiki/Petabyte

[10] http://en.wikipedia.org/wiki/Namespace

[11] http://github.com

[12] http://mobz.github.io/elasticsearch-head/

[13] http://bigdesk.org -- https://github.com/lukas-vlcek/bigdesk

[14] http://www.elastichq.org/

[15] http://www.elastichq.org/elasticsearchstats.php

[16] https://github.com/karmi/elasticsearch-paramedic

[17] https://github.com/karmi/elasticsearch-paramedic

[18] http://www.elasticsearch.org/overview/marvel/

[19] https://github.com/bleskes/sense

[20] http://couchdb.apache.org/

[21] https://github.com/elasticsearch/elasticsearch-river-twitter

[22] https://dev.twitter.com/docs/api/streaming

[23] https://dev.twitter.com/

[24] http://datasift.com/

[25] http://www.google.org/flutrends/

[26] https://twitter.com/search-advanced

[27] https://aws.amazon.com/es/ec2/?nc1=h_ls

[28] https://cloud.google.com/products/compute-engine/

[29] https://download.elasticsearch.org/kibana/kibana/kibana-3.1.0.zip

8. Annex 1

8.1 Arxiu de configuració 'elasticsearch.yml' de *Elasticsearch*

```
##################### Elasticsearch Configuration Example #####################

# This file contains an overview of various configuration settings,
# targeted at operations staff. Application developers should
# consult the guide at <http://elasticsearch.org/guide>.
#
# The installation procedure is covered at
# <http://elasticsearch.org/guide/en/elasticsearch/reference/current/setup.html>.
#
# Elasticsearch comes with reasonable defaults for most settings,
# so you can try it out without bothering with configuration.
#
# Most of the time, these defaults are just fine for running a production
# cluster. If you're fine-tuning your cluster, or wondering about the
# effect of certain configuration option, please _do ask_ on the
# mailing list or IRC channel [http://elasticsearch.org/community].

# Any element in the configuration can be replaced with environment variables
# by placing them in ${...} notation. For example:
#
# node.rack: ${RACK_ENV_VAR}

# For information on supported formats and syntax for the config file, see
# <http://elasticsearch.org/guide/en/elasticsearch/reference/current/setup-
configuration.html>


################################### Cluster ###################################

# Cluster name identifies your cluster for auto-discovery. If you're running
# multiple clusters on the same network, make sure you're using unique names.
#
cluster.name: elasticsearch_TFM_twitter


#################################### Node #####################################

# Node names are generated dynamically on startup, so you're relieved
# from configuring them manually. You can tie this node to a specific name:
#
node.name: "StopFlu_master"

# Every node can be configured to allow or deny being eligible as the master,
# and to allow or deny to store the data.
#
# Allow this node to be eligible as a master node (enabled by default):
#
node.master: true
#
# Allow this node to store data (enabled by default):
#
node.data: true

# You can exploit these settings to design advanced cluster topologies.
#
# 1. You want this node to never become a master node, only to hold data.
```

```
#    This will be the "workhorse" of your cluster.
#
# node.master: false
# node.data: true
#
# 2. You want this node to only serve as a master: to not store any data and
#    to have free resources. This will be the "coordinator" of your cluster.
#
# node.master: true
# node.data: false
#
# 3. You want this node to be neither master nor data node, but
#    to act as a "search load balancer" (fetching data from nodes,
#    aggregating results, etc.)
#
# node.master: false
# node.data: false

# Use the Cluster Health API [http://localhost:9200/_cluster/health], the
# Node Info API [http://localhost:9200/_nodes] or GUI tools
# such as <http://www.elasticsearch.org/overview/marvel/>,
# <http://github.com/karmi/elasticsearch-paramedic>,
# <http://github.com/lukas-vlcek/bigdesk> and
# <http://mobz.github.com/elasticsearch-head> to inspect the cluster state.

# A node can have generic attributes associated with it, which can later be used
# for customized shard allocation filtering, or allocation awareness. An attribute
# is a simple key value pair, similar to node.key: value, here is an example:
#
# node.rack: rack314

# By default, multiple nodes are allowed to start from the same installation location
# to disable it, set the following:
# node.max_local_storage_nodes: 3


#################################### Index ####################################

# You can set a number of options (such as shard/replica options, mapping
# or analyzer definitions, translog settings, ...) for indices globally,
# in this file.
#
# Note, that it makes more sense to configure index settings specifically for
# a certain index, either when creating it or by using the index templates API.
#
# See <http://elasticsearch.org/guide/en/elasticsearch/reference/current/index-
modules.html> and
# <http://elasticsearch.org/guide/en/elasticsearch/reference/current/indices-create-
index.html>
# for more information.

# Set the number of shards (splits) of an index (5 by default):
#
# index.number_of_shards: 5

# Set the number of replicas (additional copies) of an index (1 by default):
#
# index.number_of_replicas: 2

# Note, that for development on a local machine, with small indices, it usually
# makes sense to "disable" the distributed features:
#
# index.number_of_shards: 1
# index.number_of_replicas: 0
```

```
# These settings directly affect the performance of index and search operations
# in your cluster. Assuming you have enough machines to hold shards and
# replicas, the rule of thumb is:
#
# 1. Having more *shards* enhances the _indexing_ performance and allows to
#    _distribute_ a big index across machines.
# 2. Having more *replicas* enhances the _search_ performance and improves the
#    cluster _availability_.
#
# The "number_of_shards" is a one-time setting for an index.
#
# The "number_of_replicas" can be increased or decreased anytime,
# by using the Index Update Settings API.
#
# Elasticsearch takes care about load balancing, relocating, gathering the
# results from nodes, etc. Experiment with different settings to fine-tune
# your setup.

# Use the Index Status API (<http://localhost:9200/A/_status>) to inspect
# the index status.


#################################### Paths ####################################

# Path to directory containing configuration (this file and logging.yml):
#
# path.conf: /path/to/conf

# Path to directory where to store index data allocated for this node.
#
# path.data: /path/to/data
#
# Can optionally include more than one location, causing data to be striped across
# the locations (a la RAID 0) on a file level, favouring locations with most free
# space on creation. For example:
#
# path.data: /path/to/data1,/path/to/data2

# Path to temporary files:
#
# path.work: /path/to/work

# Path to log files:
#
# path.logs: /path/to/logs

# Path to where plugins are installed:
#
# path.plugins: /path/to/plugins


#################################### Plugin ###################################

# If a plugin listed here is not installed for current node, the node will not start.
#
# plugin.mandatory: mapper-attachments,lang-groovy


################################### Memory ####################################

# Elasticsearch performs poorly when JVM starts swapping: you should ensure that
# it _never_ swaps.
#
# Set this property to true to lock the memory:
#
```

```
# bootstrap.mlockall: true

# Make sure that the ES_MIN_MEM and ES_MAX_MEM environment variables are set
# to the same value, and that the machine has enough memory to allocate
# for Elasticsearch, leaving enough memory for the operating system itself.
#
# You should also make sure that the Elasticsearch process is allowed to lock
# the memory, eg. by using `ulimit -l unlimited`.


################################## Network And HTTP ###############################

# Elasticsearch, by default, binds itself to the 0.0.0.0 address, and listens
# on port [9200-9300] for HTTP traffic and on port [9300-9400] for node-to-node
# communication. (the range means that if the port is busy, it will automatically
# try the next port).

# Set the bind address specifically (IPv4 or IPv6):
#
# network.bind_host: 192.168.0.1

# Set the address other nodes will use to communicate with this node. If not
# set, it is automatically derived. It must point to an actual IP address.
#
# network.publish_host: 192.168.0.1

# Set both 'bind_host' and 'publish_host':
#
# network.host: 192.168.0.1

# Set a custom port for the node to node communication (9300 by default):
#
# transport.tcp.port: 9300

# Enable compression for all communication between nodes (disabled by default):
#
# transport.tcp.compress: true

# Set a custom port to listen for HTTP traffic:
#
# http.port: 9200

# Set a custom allowed content length:
#
# http.max_content_length: 100mb

# Disable HTTP completely:
#
# http.enabled: false


################################### Gateway #######################################

# The gateway allows for persisting the cluster state between full cluster
# restarts. Every change to the state (such as adding an index) will be stored
# in the gateway, and when the cluster starts up for the first time,
# it will read its state from the gateway.

# There are several types of gateway implementations. For more information, see
# <http://elasticsearch.org/guide/en/elasticsearch/reference/current/modules-
gateway.html>.

# The default gateway type is the "local" gateway (recommended):
#
# gateway.type: local
```

```
# Settings below control how and when to start the initial recovery process on
# a full cluster restart (to reuse as much local data as possible when using shared
# gateway).

# Allow recovery process after N nodes in a cluster are up:
#
# gateway.recover_after_nodes: 1

# Set the timeout to initiate the recovery process, once the N nodes
# from previous setting are up (accepts time value):
#
# gateway.recover_after_time: 5m

# Set how many nodes are expected in this cluster. Once these N nodes
# are up (and recover_after_nodes is met), begin recovery process immediately
# (without waiting for recover_after_time to expire):
#
# gateway.expected_nodes: 2


############################### Recovery Throttling ###############################

# These settings allow to control the process of shards allocation between
# nodes during initial recovery, replica allocation, rebalancing,
# or when adding and removing nodes.

# Set the number of concurrent recoveries happening on a node:
#
# 1. During the initial recovery
#
# cluster.routing.allocation.node_initial_primaries_recoveries: 4
#
# 2. During adding/removing nodes, rebalancing, etc
#
# cluster.routing.allocation.node_concurrent_recoveries: 2

# Set to throttle throughput when recovering (eg. 100mb, by default 20mb):
#
# indices.recovery.max_bytes_per_sec: 20mb

# Set to limit the number of open concurrent streams when
# recovering a shard from a peer:
#
# indices.recovery.concurrent_streams: 5


################################## Discovery ##################################

# Discovery infrastructure ensures nodes can be found within a cluster
# and master node is elected. Multicast discovery is the default.

# Set to ensure a node sees N other master eligible nodes to be considered
# operational within the cluster. Its recommended to set it to a higher value
# than 1 when running more than 2 nodes in the cluster.
#
# discovery.zen.minimum_master_nodes: 1

# Set the time to wait for ping responses from other nodes when discovering.
# Set this option to a higher value on a slow or congested network
# to minimize discovery failures:
#
# discovery.zen.ping.timeout: 3s

# For more information, see
```

```
# <http://elasticsearch.org/guide/en/elasticsearch/reference/current/modules-
discovery-zen.html>

# Unicast discovery allows to explicitly control which nodes will be used
# to discover the cluster. It can be used when multicast is not present,
# or to restrict the cluster communication-wise.
#
# 1. Disable multicast discovery (enabled by default):
#
# discovery.zen.ping.multicast.enabled: false
#
# 2. Configure an initial list of master nodes in the cluster
#    to perform discovery when new nodes (master or data) are started:
#
# discovery.zen.ping.unicast.hosts: ["host1", "host2:port"]

# EC2 discovery allows to use AWS EC2 API in order to perform discovery.
#
# You have to install the cloud-aws plugin for enabling the EC2 discovery.
#
# For more information, see
# <http://elasticsearch.org/guide/en/elasticsearch/reference/current/modules-
discovery-ec2.html>
#
# See <http://elasticsearch.org/tutorials/elasticsearch-on-ec2/>
# for a step-by-step tutorial.

# GCE discovery allows to use Google Compute Engine API in order to perform
discovery.
#
# You have to install the cloud-gce plugin for enabling the GCE discovery.
#
# For more information, see <https://github.com/elasticsearch/elasticsearch-cloud-
gce>.

# Azure discovery allows to use Azure API in order to perform discovery.
#
# You have to install the cloud-azure plugin for enabling the Azure discovery.
#
# For more information, see <https://github.com/elasticsearch/elasticsearch-cloud-
azure>.

################################## Slow Log ##################################

# Shard level query and fetch threshold logging.

#index.search.slowlog.threshold.query.warn: 10s
#index.search.slowlog.threshold.query.info: 5s
#index.search.slowlog.threshold.query.debug: 2s
#index.search.slowlog.threshold.query.trace: 500ms

#index.search.slowlog.threshold.fetch.warn: 1s
#index.search.slowlog.threshold.fetch.info: 800ms
#index.search.slowlog.threshold.fetch.debug: 500ms
#index.search.slowlog.threshold.fetch.trace: 200ms

#index.indexing.slowlog.threshold.index.warn: 10s
#index.indexing.slowlog.threshold.index.info: 5s
#index.indexing.slowlog.threshold.index.debug: 2s
#index.indexing.slowlog.threshold.index.trace: 500ms

################################## GC Logging ################################

#monitor.jvm.gc.young.warn: 1000ms
#monitor.jvm.gc.young.info: 700ms
```

```
#monitor.jvm.gc.young.debug: 400ms

#monitor.jvm.gc.old.warn: 10s
#monitor.jvm.gc.old.info: 5s
#monitor.jvm.gc.old.debug: 2s
```

8.2 Arxiu de configuració 'config.js' de *Kibana*

```
/** @scratch /configuration/config.js/1
 *
 * == Configuration
 * config.js is where you will find the core Kibana configuration. This file contains
parameter that
 * must be set before kibana is run for the first time.
 */
define(['settings'],
function (Settings) {


  /** @scratch /configuration/config.js/2
   *
   * === Parameters
   */
  return new Settings({

    /** @scratch /configuration/config.js/5
     *
     * ==== elasticsearch
     *
     * The URL to your elasticsearch server. You almost certainly don't
     * want +http://localhost:9200+ here. Even if Kibana and Elasticsearch are on
     * the same host. By default this will attempt to reach ES at the same host you
have
     * kibana installed on. You probably want to set it to the FQDN of your
     * elasticsearch host
     *
     * Note: this can also be an object if you want to pass options to the http
client. For example:
     *
     *  +elasticsearch: {server: "http://localhost:9200", withCredentials: true}+
     *
     */
    elasticsearch: "http://192.168.0.197:9200",

    /** @scratch /configuration/config.js/5
     *
     * ==== default_route
     *
     * This is the default landing page when you don't specify a dashboard to load.
You can specify
     * files, scripts or saved dashboards here. For example, if you had saved a
dashboard called
     * `WebLogs' to elasticsearch you might use:
     *
     * default_route: '/dashboard/elasticsearch/WebLogs',
     */
    default_route     : '/dashboard/file/default.json',

    /** @scratch /configuration/config.js/5
```

```
     *
     * ==== kibana-int
     *
     * The default ES index to use for storing Kibana specific object
     * such as stored dashboards
     */
    kibana_index: "kibana-int",

    /** @scratch /configuration/config.js/5
     *
     * ==== panel_name
     *
     * An array of panel modules available. Panels will only be loaded when they are
defined in the
     * dashboard, but this list is used in the "add panel" interface.
     */
    panel_names: [
      'histogram',
      'map',
      'goal',
      'table',
      'filtering',
      'timepicker',
      'text',
      'hits',
      'column',
      'trends',
      'bettermap',
      'query',
      'terms',
      'stats',
      'sparklines'
    ]
  });
});
```