

Trabajo Fin de Programa. Experto en Big Data

Análisis en tiempo real de las respuestas a invitaciones en Meetup

Autor: Pedro Peña García

Tutor: Iván de Prado Alonso

ÍNDICE

[Objetivos](#)

[Arquitectura](#)

[Hadoop / Cluster Yarn](#)

[Cluster Elasticsearch](#)

[Apache Kafka + Zookeeper](#)

[Servidor Redis](#)

[Software Desarrollado](#)

[Procesamiento en tiempo real](#)

[Enriquecimiento de datos](#)

[Analítica Avanzada](#)

[Predicción de respuestas de confirmación a eventos](#)

[Clustering de Grupos](#)

[Visualización](#)

[RealTime.](#)

[Dashboards](#)

[Listado de Grupos](#)

[Listado de Eventos](#)

[Alertas](#)

[Trending Groups](#)

[Groups Clusters](#)

[Conclusiones](#)

Objetivos

El objetivo de este proyecto consiste en procesar, analizar y explotar la información provista por la plataforma de eventos Meetup, mediante el uso de tecnologías Big Data.

Meetup es una plataforma de redes sociales creado por Scott Heiferman, Matt Meeker y Peter Kamali en el año 2002. Meetup permite a sus miembros reunirse en la vida real a través de grupos unidos por un interés común como política, deporte, cultura, etc. Meetup publica en tiempo real, mediante streaming, las respuestas a las invitaciones a los diferentes grupos y además ofrece un API REST. http://www.meetup.com/es-ES/meetup_api/ para obtener información sobre grupos y eventos.

Haciendo uso tanto del canal de stream, como del API proporcionado por Meetup, se ha construido una plataforma basada en una arquitectura lambda

[https://en.wikipedia.org/wiki/Lambda_architecture] Definida como un conjunto de principios para crear arquitecturas de sistemas Big Data en tiempo real. Mediante la cual:

- Se Obtiene, enriquece, analiza y almacena información en tiempo real (*near real time*) [**Speed Layer**]
- Se realiza un análisis avanzado sobre los datos capturados mediante el uso de algoritmos de machine learning [**Batch Layer**]
- Se presenta la información obtenida en los procesos anteriores mediante una aplicación web [**Serving Layer**]

Específicamente, la plataforma construida aborda y proporciona solución en las siguientes áreas:

- **Procesamiento en tiempo real del flujo de entrada respuestas en Meetup.** Procesar, detectar y notificar mediante sistema de alertas, aquellos eventos que reciben un gran número de respuestas por minuto, así como la obtención de los eventos “trending” con mayor número de respuestas en cada uno de los países.
- **Enriquecimiento de los datos.** Añadir información a las respuestas recibidas, a través del API de Meetup u otras fuentes externas con la idea de poder realizar un mejor análisis de los datos.
- **Análisis y procesamiento batch de la información obtenida.** Obtener resultados relevantes de los datos procesados, así como poner en práctica algunos de los algoritmos de machine learning que se han impartido en el transcurso del curso.
- **Visualización de datos.** Visualizar mediante una aplicación web los datos que vamos recibiendo en tiempo real, así como el resultado de los diferentes análisis que se realicen posteriormente sobre ellos.

Arquitectura

Con el propósito de alcanzar los objetivos marcados, las herramientas y frameworks que se han utilizado en este proyecto y sobre las que se ha construido la arquitectura, son las siguientes:

- **Docker.** Proyecto de código abierto, que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización a nivel de sistema operativo en Linux.
<https://www.docker.com/>
- **Apache Hadoop.** Framework de software libre que soporta aplicaciones distribuidas. Como arquitectura cluster sobre la cual lanzar los procesos/aplicaciones spark.
<http://hadoop.apache.org/>
- **Spark & Spark Streaming.** en su versión 1.6.1, sobre scala, para realizar la captura, procesamiento y análisis de los datos. <http://spark.apache.org/>
También se hará uso de la librería [MLlib](#). La cual permite la utilización de diferentes algoritmos de machine learning de manera sencilla y escalable.
- **Elastic Search 2.3.3.** Servidor de búsqueda basado en Lucene. Provee un motor de búsqueda de texto completo, distribuido y con capacidad de multi-tenencia con una interfaz web RESTful. Utilizado como nuestro sistema de almacenamiento de datos
<https://www.elastic.co/products/elasticsearch>
- **Kibana 4.5.3.** Sistema de visualización open source para datos almacenados en elasticsearch. <https://www.elastic.co/products/kibana>
- **Grafana 3.1.0.** Aplicación web open source para la creación de analytics dashboards. <http://grafana.org/>
- **Apache Kafka.** Sistema de mensajería productor/suscriptor. Utilizado para comunicar aplicaciones spark en el proceso de enriquecimiento de los datos de entrada. <http://kafka.apache.org/>
- **Apache Zookeeper.** Servicio de configuración centralizada y registro de nombres. <https://zookeeper.apache.org/> . Instalación necesaria como requerimiento de apache kafka
- **Redis.** Sistema de almacenamiento de datos en memoria, que puede usarse como base de datos, caché o broker de mensajería. <http://redis.io/> En nuestro caso se ha utilizado como cache
- **Apache Zeppelin.** Implementación del concepto de web notebook, centrado en la analítica de datos interactivo mediante lenguajes y tecnologías como Shell, Spark, SparkSQL, Hive, Elasticsearch, R, etc... <https://zeppelin.apache.org/>

En la siguiente figura se muestra la arquitectura del proyecto con los distintos elementos que la componen.

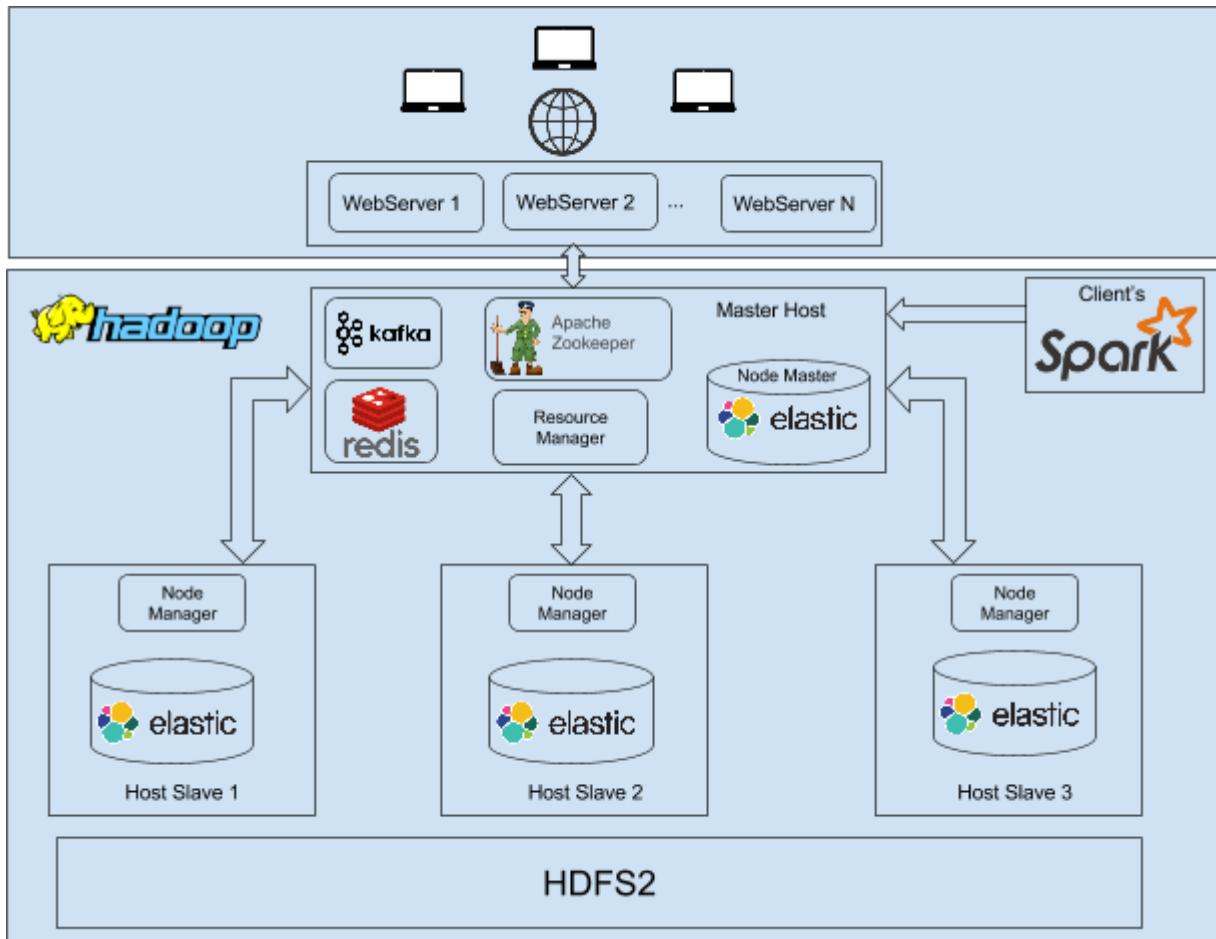


Figura-1. Esquemático Arquitectura.

Hadoop / Cluster Yarn

La arquitectura se ha construido sobre un cluster hadoop / yarn sobre contenedores docker. El cluster está compuesto por cuatro máquinas, si bien es escalable horizontal y verticalmente. La configuración que se muestra en la [Figura-2](#), consta de una máquina máster (Resource Manager y Name Node) y tres máquinas esclavas (Node Managers) donde se ejecutarán las tareas enviadas a yarn.

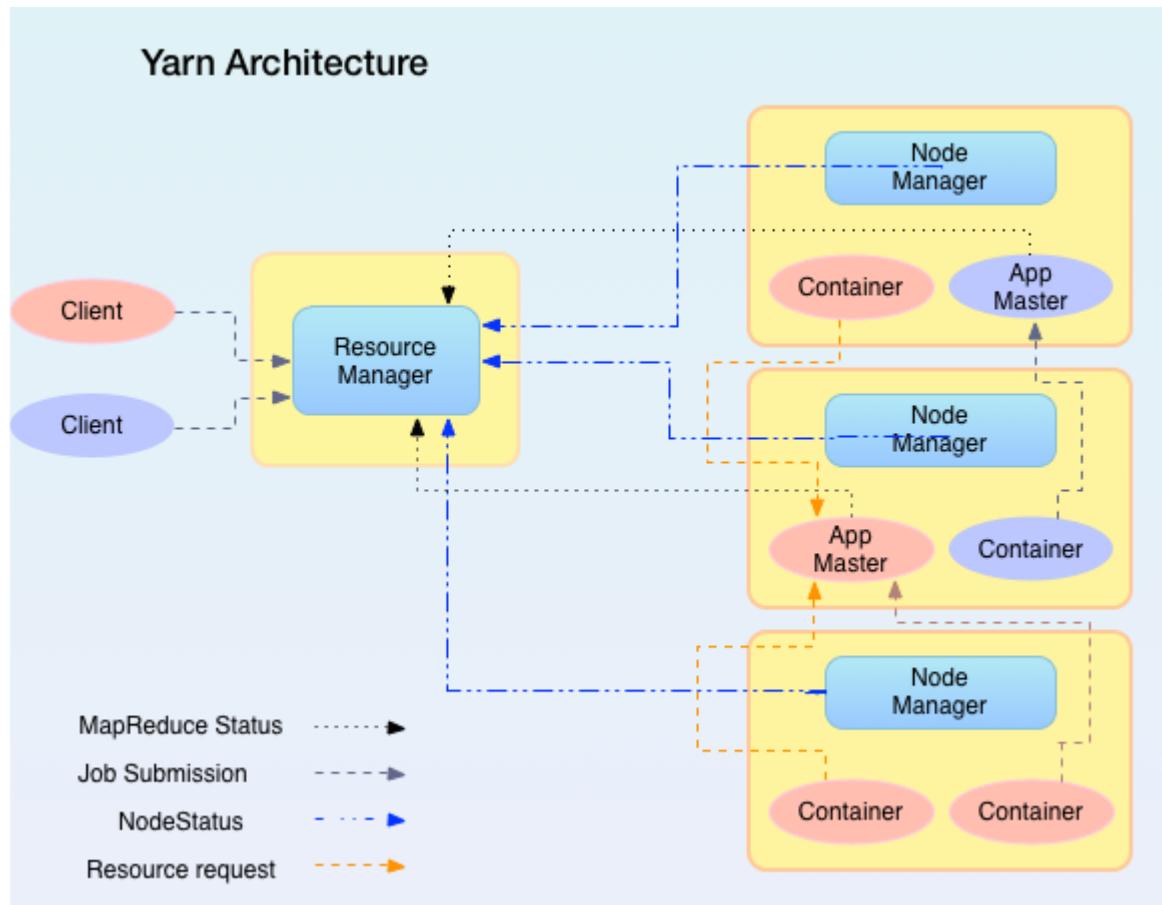


Figura-2. Esquemático arquitectura cluster hadoop/yarn

Los elementos que intervienen son:

- **Resource Manager.** Provee un endpoint a los clientes para hacer la solicitud de trabajos. Tiene **ApplicationsManager** incorporados que gestionan los trabajos en el cluster.
- **Node Manager.** Hay uno por nodo esclavo, es el responsable de la monitorización y gestión de los recursos. Recoge las directrices del Resource Manager y crea contenedores basado en los requerimientos de la tarea.
- **Application Master.** Se despliega junto al Node Manager. Es creado por Job y controla la monitorización y la ejecución de las tareas usando el contenedor. Negocia los requerimientos de los recursos para el Job con el ResourceManager y tiene la responsabilidad de completar las tareas. proporciona la tolerancia a fallos a nivel de tarea.
- **Container.** Es la unidad básica de la asignación en lugar de un Map o Reduce slot en Hadoop 1.x. El contenedor se define con atributos como la memoria, CPU, disco etc... aplicaciones como procesamiento gráfico y MPI.

Cluster Elasticsearch

En cada una de las máquinas del cluster descrito en el punto anterior, se ha instalando un nodo elasticsearch; proveyendo de este modo a la arquitectura de un cluster elasticsearch.

Lo ideal en una arquitectura real sería utilizar máquinas diferentes para montar este cluster, sin embargo por limitación de recursos en este proyecto se han utilizado las mismas sobre las que se instaló el cluster yarn.

Sobre este cluster se han creado tres índices en los que se almacenan de forma separada las respuestas, grupos y eventos procesadas en las distintas aplicaciones implementadas y que se detallarán más adelante.

Como herramienta de gestión / monitorización del cluster, se ha utilizado el plugin elasticHQ <http://www.elastichq.org/>

Apache Kafka + Zookeeper

Con el objetivo de comunicar la aplicación web con los procesos de spark streaming así como para gestionar el proceso de enriquecimiento de las respuestas de meetup recibidas, se ha instalado en el host master, Apache Kafka + Apache Zookeeper. Dichos procesos se explicarán detalladamente más adelante.

Servidor Redis

Por último se ha instalado también en el host master, un servidor redis. Dicho servidor tiene como objeto servir de caché de la aplicación web, utilizada para servir el contenido que visualizarán los usuarios finales. Y al mismo tiempo como herramienta para almacenar parámetros de configuración que alguna de las aplicaciones spark implementadas como se explicará más adelante.

Software Desarrollado

Una vez configurada e construida la arquitectura sobre la que llevar a cabo el proyecto, el siguiente paso ha sido desarrollar el código y aplicaciones software para poder realizar la captura, procesamiento, almacenamiento y análisis de los datos que nos marcamos como objetivo.

Para ello se ha decidido utilizar el lenguaje de programación [scala](#), el cual también ha sido utilizado en el transcurso del curso y para el que spark provee API de programación.

A continuación se detallan los desarrollos realizados en cada una de las áreas marcadas como objetivos

Procesamiento en tiempo real

Se ha implementado una aplicación mediante Spark Streaming

<https://spark.apache.org/docs/latest/streaming-programming-guide.html> que se conecta vía http a la fuente de entrada de Meetup. <http://stream.meetup.com/2/rsvps> para obtener las respuestas que se van produciendo. Este flujo streaming de entrada:

1. Es almacenado en elasticsearch en un índice con nombre **meetup-rsvps**, haciendo uso del API elasticsearch para hadoop spark.
<https://www.elastic.co/guide/en/elasticsearch/hadoop/current/spark.html>
2. Se envía a un topic kafka para que se realice el proceso de enriquecimiento. Se ha tomado esta decisión para desacoplar el proceso de enriquecimiento, lo cual nos permite por un lado, no hacer más pesado el proceso de captura de los eventos, y al mismo tiempo permitir escalar horizontalmente el proceso de enriquecimiento, en caso de tener gran número de respuestas de entrada.
3. Se realizan análisis en tiempo real sobre el flujo de entrada, para ello se ha hecho uso de las operaciones window que provee spark streaming
<http://spark.apache.org/docs/latest/streaming-programming-guide.html#window-operations>. Que permiten realizar transformaciones a los datos de entrada contenidos en una ventana de tiempo. Tal como refleja en la [Figura-3](#).

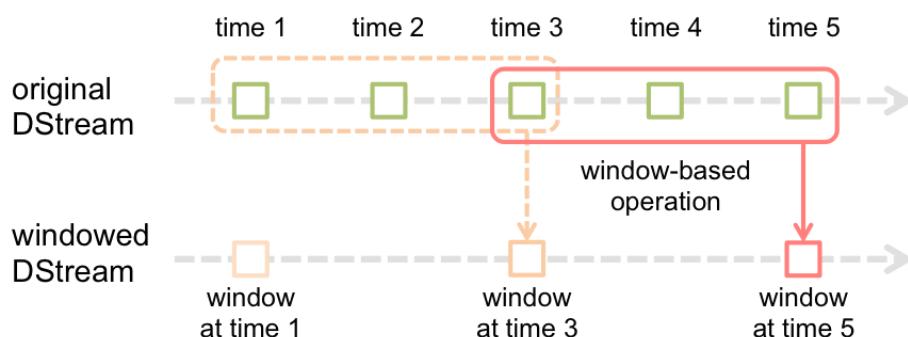


Figura-3. Esquemático Window Operation

En nuestro caso realizamos dos tipos de análisis mediante estas transformaciones:

- a. Mediante una ventana de tiempo de un minuto de duración. Se cuentan las respuestas de entrada por cada evento. De manera que para cada evento que supere un umbral de respuestas configurado mediante Redis, se envía una alerta a un topic kafka configurado a tal efecto. Este topic es consumido por una aplicación web que se encargará de mostrar estas alertas al usuario final.
- b. Con otra ventana de tiempo, esta vez de una hora. Se cuentan las respuestas recibidas por evento y se agrupan por país. Cada minuto este cómputo es enviado a un topic kafka que es consumido por una aplicación web.

Enriquecimiento de datos

Como se indicaba en el segundo punto del apartado anterior. Se ha utilizado un topic kafka, al cual se envían peticiones de enriquecimiento para cada respuesta recibida. Este topic es consumido por una aplicación Spark Streaming, la cual se encarga de realizar el proceso de enriquecimiento de los datos, que consta de tres operaciones.

1. Las respuestas recibidas por la fuente streaming de meetup, contienen información del grupo al que pertenece el evento, pero dicha información se reduce prácticamente al id y nombre del grupo, por lo que se ha decidido hacer uso del API de Meetup para obtener toda la información de los grupos y almacenarla en elasticsearch en un índice bajo el nombre **meetup-groups**, con el objetivo de poder explotar esta información mediante gráficas e informes en la parte de visualización del proyecto.

Toda la información sobre el método del API Meetup para consulta de grupos, se encuentra en el siguiente enlace.

https://secure.meetup.com/meetup_api/console/?path=:urlname

2. También podemos encontrar información sobre el evento al que pertenecen las respuestas. Por la misma razón que en el caso anterior, se ha decidido obtener toda la información del evento ofrecida por Meetup mediante su API y almacenarla en elasticsearch en el índice de nombre **meetup-events**.

Entre la información asociada al evento, se encuentra el número de personas que confirmaron la asistencia al mismo, dato que se ha utilizado posteriormente en la fase de análisis, para predecir el número de confirmaciones que recibirán futuros eventos.

Se puede consultar toda la información del método para consulta de eventos del API Meetup en el siguiente enlace.

https://secure.meetup.com/meetup_api/console/?path=:urlname/events/:id

3. Por último y no menos importante, aprovechando la información obtenida del grupo al que pertenece cada respuesta se ha obtenido la zona horaria a partir del cual podemos calcular la hora y fecha local, así como el día de la semana en el que se

produce cada respuesta.

Esto ha sido de gran ayuda, tanto a la hora de realizar informes de visualización, como a la hora de realizar análisis sobre los datos. Ya que nos permiten por ejemplo saber, que horas del día, días de la semana, etc... Son las que más actividad tienen, en cuanto a eventos celebrados y respuestas a invitaciones. Esto se mostrará en detalle en la parte de visualización, no obstante y a modo de ejemplo la [Figura-4](#), muestra el número de respuestas según la hora del día y en la [Figura-5](#), se puede ver el número de respuestas según el día de la semana

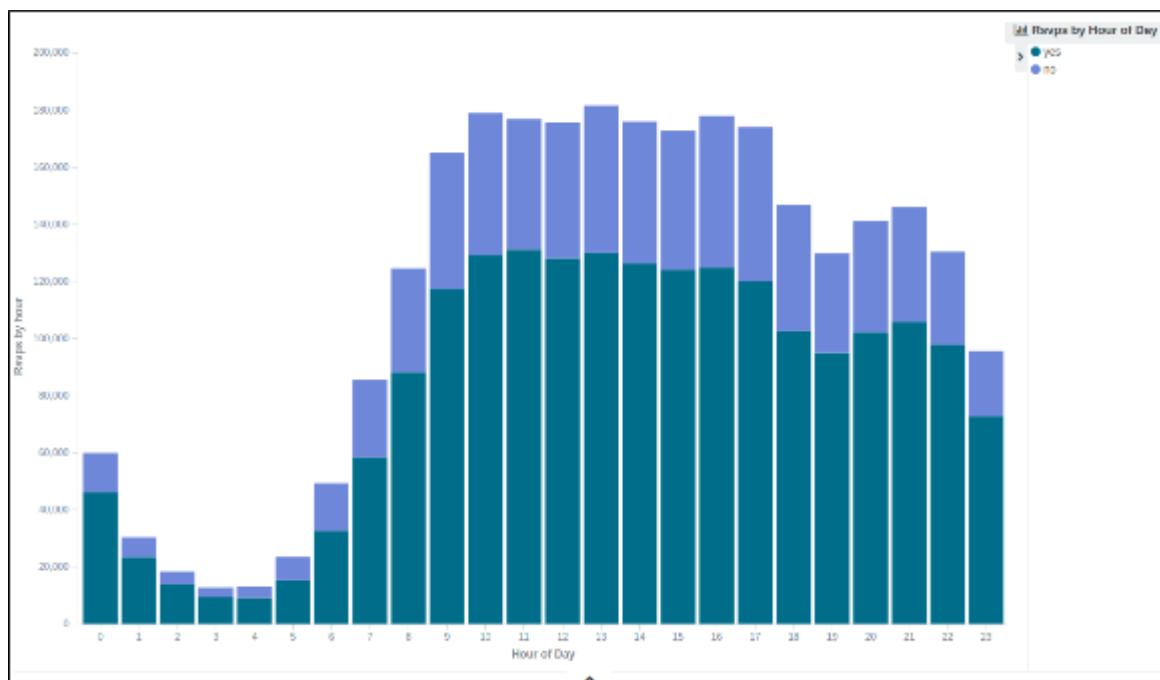


Figura-4. Número de respuestas según hora del día

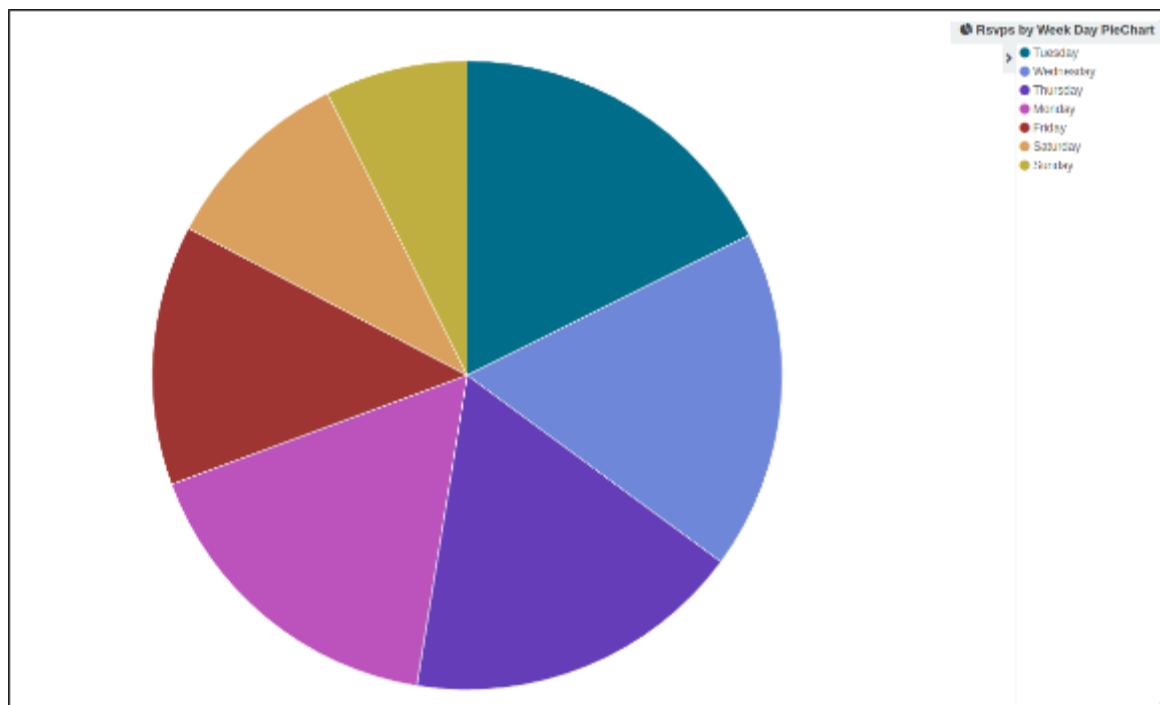


Figura-5. Número de respuestas según día de la semana

Cabe destacar por último que para el correcto almacenamiento de los datos en Elasticsearch ha sido necesario configurar el mapeo de los campos para cada índice, para configurar por ejemplo que campos eran de tipo date, cuales eran geoposiciones, aquellos que no se quieren analizar, etc.. Toda la info sobre el mapeo de campos en Elasticsearch se puede encontrar en el siguiente enlace.

<https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping.html>

Analítica Avanzada

El propósito de esta parte del proyecto es la utilización de técnicas de analítica avanzada, con un doble objetivo. En primer lugar la utilización y estudio de las distintas posibilidades que ofrece la librería de machine learning MLlib desarrollada para Spark. En segundo lugar, la utilización de este tipo de técnicas nos permitirá obtener modelos que nos ayuden a entender mejor la naturaleza de los eventos capturados en las fases previas, siendo uno de los elementos fundamentales sobre los que se apoyará la parte de visualización.

No es objeto de este trabajo, explicar los detalles y particularidades de las distintas técnicas de machine learning, aunque resulta conveniente una breve introducción.

Dentro de las distintas técnicas de machine learning, se pueden distinguir dos grandes grupos de técnicas, las técnicas de aprendizaje supervisadas y las no supervisadas. Las técnicas supervisadas se caracterizan por requerir un conjunto de datos que deben estar etiquetados a priori, es decir, en los que se conoce a priori el valor de la/s variable/s de interés. Dicho conjunto de datos está formado por un conjunto de características (variables de entrada) y una o varias características (variables de salida). Atendiendo a la naturaleza de las variables de salida, encontramos problemas de regresión, en los que la variable de salida es un valor numérico y problemas de clasificación en los que el valor de la variable de salida corresponde con una etiqueta (enfermo/sano). A partir de dicho conjunto de datos, los distintos algoritmos crean modelos con la capacidad de realizar una previsión/clasificación para un nuevo conjunto de variables de entrada. Por su parte las técnicas no supervisadas, no requieren que el conjunto de los datos estén etiquetados, siendo su objetivo la búsqueda de patrones ocultos dentro del conjunto de datos, este tipo de técnicas tiene entre sus representantes más significativos las técnicas de clustering o agrupamiento y las técnicas de reducción de dimensionalidad como PCA (Principal component analysis).

Un aspecto fundamental es el modo en el que se crean estos modelos, idealmente los modelos deberían crearse de modo dinámico a medida que se captan nuevos eventos, de hecho la librería MLlib cuenta con algunos algoritmos orientados a tal efecto como

<http://spark.apache.org/docs/latest/ml-clustering.html#streaming-k-means>, sin embargo la utilización de este tipo de técnicas de modo dinámico, es un trabajo que se ha quedado fuera del alcance del proyecto por las limitaciones de tiempo y a fin de lograr un equilibrio entre todas las partes del mismo. Por ello se optado por una implementación de procesos batch, que nos permitan obtener información de interés, sobre los datos que se han capturado, procesado y enriquecido en las fases anteriores. De manera que podamos a posteriori, mostrar las conclusiones obtenidas con estos análisis, en la fase de visualización.

Se han utilizado dos técnicas distintas de machine learning, una de aprendizaje supervisado y otra de aprendizaje no supervisado, cada una de ellas con los siguientes objetivos:

- Modelo de aprendizaje supervisado, planteado como un problema de regresión. Obtener un sistema de predicción sobre el número de respuestas de confirmación que tendrá un evento.
- Modelo de aprendizaje no supervisado, cuyo objetivo el agrupamiento de los distintos grupos, en base a los topics/intereses que los componen.

Predicción de respuestas de confirmación a eventos

Para tal propósito se han probado las implementaciones de los modelos predictivos sobre Árboles de Decisión y Random Forest. Toda la información referente a los mismos se puede encontrar en la página de Spark. <http://spark.apache.org/docs/latest/mllib-decision-tree.html>, <http://spark.apache.org/docs/latest/mllib-ensembles.html#random-forests>

Un árbol de decisión, al igual que el resto de las técnicas de aprendizaje supervisado, parte de un conjunto de variables de entrada, a partir de las cuales se construye un modelo que intenta predecir una salida. En nuestro caso la variable de salida, es el número de confirmaciones de asistencia a un evento, las variables de entrada se detallan a continuación. En el caso de los árboles de decisión, el modelo construido utiliza las distintas variables de entrada para tomar una decisión sobre cuál es el valor de la variable de salida.

A continuación se muestra un ejemplo de árbol de decisión para determinar si un pasajero sobrevivió o no a bordo del Titanic (variable de salida), utilizando el sexo del pasajero, la edad y el número de hermanos y esposos que también iban a bordo del Titanic (variables de entrada).

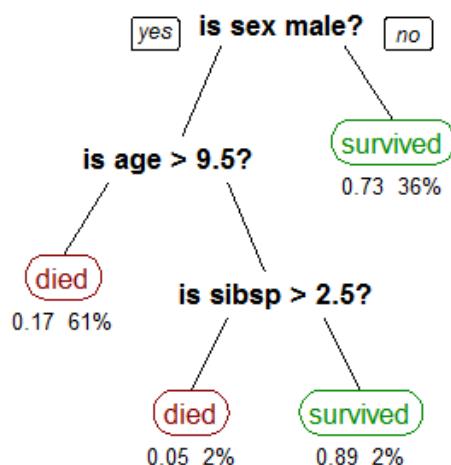


Figura-6. Ejemplo árbol de decisión

En nuestro caso de entre los datos disponibles en la información del evento los atributos de entrada elegidos han sido:

- **yes_rsvp_count**: Número de confirmaciones de asistencia
- **month(time)**: Mes en el que se celebró el evento
- **week_day**: Día de la semana en el que se celebró el evento
- **local_hour**: Hora local en la que se celebró el evento
- **group.city**: Ciudad donde se celebró el evento
- **group.country**: País en el que se celebró el evento
- **group.members**: Número de miembros del grupo
- **group.category.id**: Id de la categoría del evento

Un ejemplo de conjunto de entrada de datos sería:

```
(12.0,[7.0,5.0,19.0,448.0,72.0,1066.0,26.0])
(13.0,[7.0,2.0,13.0,561.0,72.0,312.0,31.0])
(11.0,[7.0,0.0,19.0,1784.0,72.0,61.0,14.0])
(11.0,[7.0,4.0,21.0,1086.0,72.0,1228.0,31.0])
(11.0,[7.0,2.0,10.0,64.0,72.0,486.0,3.0])
(11.0,[7.0,6.0,14.0,1387.0,72.0,564.0,21.0])
(13.0,[7.0,0.0,21.0,66.0,72.0,1573.0,34.0])
(14.0,[7.0,2.0,12.0,1183.0,72.0,312.0,34.0])
(14.0,[7.0,0.0,11.0,1784.0,72.0,1685.0,23.0])
```

Como se puede apreciar todos los valores son numéricos y entre ellos se encuentran la ciudad y el país del evento. Esto se debe a que el modelo de entrada sólo soporta valores numéricos. Por lo tanto se ha tenido que realizar una transformación de String a Number mediante el uso de mapas, sirva como ejemplo:

```
(cities,Map(Talent -> 408, South Lake Tahoe -> 650, Sycamore -> 1555, Yorba Linda ->
1791, Amsterdam -> 657, Clayton -> 211,...))
```

Los modelos, una vez entrenados con un conjunto de datos de entrada, pueden predecir el número de confirmaciones de asistencia que tendrá un evento.

En las pruebas, se han utilizado datos de entrada, con eventos de hasta un año, dejando el 70% como datos de entrenamiento y el 30% restante como datos de test.

A modo de ejemplo mostramos algunos de los resultados obtenidos, donde el valor de la izquierda representa el número de confirmaciones real y el de la derecha, la predicción realizada por el modelo.

Resultados con Árboles de Decisión

```
(18,16)(55,66)(12,27)(27,37)(42,36)(16,25)(28,27)(47,16)(14,18)(12,21)(18,18)(16,19)(12,17)(16,17)
(13,11)(18,17)(17,21)(98,24)(12,21)(21,21)(24,31)(11,45)(51,18)(12,32)(17,17)(16,37)(14,15)(12,17)
(16,15)(28,18)(12,25)(28,28)(14,27)(28,33)(11,22)(14,27)(12,17)(14,16)(11,15)(12,40)(28,17)(32,20)
```

Resultados con Random Forest

```
(19,48)(12,72)(13,18)(11,20)(11,23)(16,41)(14,53)(14,24)(14,32)(14,25)(14,19)(43,31)(15,19)(40,31)
(12,19)(13,17)(26,20)(16,19)(12,18)(14,20)(14,23)(19,20)(14,55)(13,25)(11,29)(12,19)(16,26)(31,19)
(15,17)(22,25)(14,16)(11,40)(14,24)(59,20)(13,45)(16,21)(16,29)(13,16)(15,19)(30,37)(11,20)((41,40)
```

Como se puede apreciar no son unas predicciones muy acertadas, pero teniendo en cuenta la variabilidad de los datos y los pocos atributos de entrada con los que contamos, el problema encierra una gran dificultad.

En cualquier caso, el objetivo de construir la arquitectura mediante la cual, ser capaz de realizar análisis sobre grandes cantidades de datos, ha sido conseguido. Como líneas futuras de desarrollo estaría el reto de conseguir un predictor con mayor fiabilidad.

Cabe destacar al mismo tiempo alguna de las consideraciones realizadas durante el proceso de análisis.

Tal y como muestra la [Figura-7](#) en un primer análisis sobre los datos de los eventos, se ha observado que existe una gran cantidad de eventos para los que se reciben menos de diez confirmaciones. Este hecho, dificulta la obtención de un modelo de regresión que sea capaz de realizar predicciones precisas. A modo de ejemplo, un modelo predictivo que diera como resultado siempre un 1, tendría unos buenos resultados de error, ya que del número total de eventos, existe un alto porcentaje a los que solo confirman su asistencia un único participante, tal y como se observa en la [Figura-7](#).

Al entrenar y predecir el modelo con estos datos, los resultados de las predicciones obtenidos eran muy inferiores a la realidad, por lo tanto se ha decidido entrenar el modelo, solo con eventos que tienen más de diez confirmaciones, como resultado el modelo nunca predice un número de asistentes menor de diez, pero a cambio los resultados obtenidos eran bastante más cercanos a la realidad.

Otra opción que se ha contemplado para solucionar este problema, es hacer un primer modelo de clasificación que determine, si el evento recibiría más o menos de diez respuestas y en función de esta primera clasificación, intentar predecir el número de confirmaciones, teniendo para ello dos modelos, uno entrenado con eventos para los que se han recibido menos de diez confirmaciones y otro para eventos con más de diez. Pero no ha sido posible llevarlo a cabo en los plazos disponibles para la entrega del proyecto. Finalmente y continuando con esta última línea de trabajo propuesta, se podría tratar el problema de predecir el número de asistentes como un problema de clasificación en lugar de regresión, para ello se podrían clasificar los eventos en distintas clases, de baja participación (<10 asistentes), de media participación (>10 y <100) y de alta participación (>100), de tal modo que el modelo de diera una predicción exacta del número de participantes sino un rango aproximado.

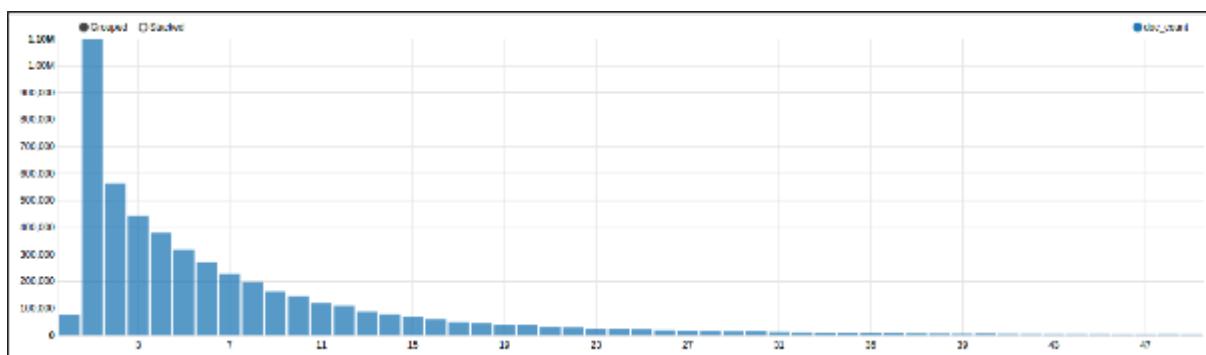


Figura-7. Número de grupos según las respuestas de confirmación de asistencia

Por otra parte se ha contemplado la posibilidad de incorporar algún atributo más a los datos de entrada como podría ser el número de habitantes de la ciudad, pero tras ver la gran variabilidad y poca correlación de los datos, se ha descartado la opción.

Se llegaron a encontrar eventos del mismo grupo para los que el número de confirmaciones difería a nivel de centenas, lo cual dificulta enormemente la posibilidad de predecir.

Destacar que observando los árboles resultado del proceso de entrenamiento, las “features/características” que ocupaban las ramas más altas y por lo tanto más importantes a la hora de clasificar, han sido. La ciudad, la categoría y el número de miembros del grupo.

Por último comentar que el modelo utilizado finalmente como sistema de predicción del proyecto, ha sido el implementado mediante el algoritmo random forest.

Para ello nos hemos basado en el Error cuadrático medio ([ECM](#)) de las predicciones de ambos modelos. Los resultados obtenidos en los test han obtenido un menor ECM para las predicciones con el modelo Random Forest

El ECM mide el promedio de los errores al cuadrado, es decir, la diferencia entre el valor estimado y lo que se estima.

$$ECM = \frac{\sum_{i=1}^n (e_i)^2}{n}$$

Figura-8. Fórmula error cuadrático medio

Clustering de Grupos

Para realizar la clasificación de grupos se ha utilizado el algoritmo K-Means implementado por la librería mllib de Spark. <http://spark.apache.org/docs/latest/mllib-clustering.html>

El objetivo de este algoritmo, es particionar un conjunto de n observaciones en k grupos, en el que cada observación pertenece al grupo más cercano a la media

Dado un conjunto de observaciones ($\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$), donde cada una, es un vector real de d dimensiones, k-means construye una partición de las observaciones en k conjuntos ($k \leq n$) a fin de minimizar la suma de los cuadrados dentro de cada grupo (WCSS): $\mathbf{S} = \{S_1, S_2, \dots, S_k\}$

$$\arg \min_{\mathbf{S}} \sum_{i=1}^k \sum_{\mathbf{x}_j \in S_i} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2$$

donde $\boldsymbol{\mu}_i$ es la media de puntos en S_i .

En nuestro caso las observaciones son cada uno de los grupos y para crear el vector que define cada grupo, se ha utilizado la lista de topics de cada a cada grupo

Por ejemplo, el grupo Johnstown Region Startup Meetup. Presenta la siguiente lista de topics: smallbiz, prodev, professional-networking, business-strategy, entrepreneurship, business-entrepreneur-networking, startup-businesses.

Para pasar la lista de topics de cada grupo a un vector. El primer paso ha sido obtener la lista de todos los topics que existen en el conjunto de datos, asignando a cada topic una posición en el vector. De esta manera cada vector tendrá como longitud el número total de topics existentes y contendrá el valor 1 en aquellas posiciones que correspondan a cada uno de los topics del grupo.

Por ejemplo teniendo el conjunto de topics: [social, sports, friends, movies, business]

Un grupo cuyos topics son: sports, friends. Sería traducido como el vector: (0,1,1,0,0)

Una vez tenemos todos los vectores que representan cada grupo, el algoritmo clasifica cada uno en un cluster.

A modo de ejemplo. Imaginemos un cluster compuesto por tres grupos. C1 -> G1, G2, G3

G1: topics (A,B,C)

G2: topics (B,C,D)

G3: topics (B,C,E,)

Los topics del cluster resultan. (A,B,C,B,C,D,B,C,E,)

A partir de esto se ha elaborado un nube de etiquetas con el objetivo de comprobar si la agrupación realizada por el algoritmo K-Mean tiene cierto sentido.

Tal y como se puede apreciar en la [Figura-9](#), parece que la agrupación realizada presenta resultados que están íntimamente relacionados.

Para realizar estas nubes de etiqueta se ha hecho uso de la siguiente librería python

https://github.com/amueller/word_cloud



Figura-9. Nube de etiquetas con los topic de los grupos de cada cluster

Como se puede apreciar en la imagen anterior, el algoritmo ha sido capaz de agrupar de modo automático grupos que están íntimamente relacionados, como grupos orientados al desarrollo de software y la tecnología, grupos de idiomas o grupos orientados al desarrollo empresarial.

Destacar por último, que se ha realizado una pequeña modificación de la implementación del algoritmo K-Means de mllib, para poder obtener la distancia de cada grupo al centroide (valor medio de cada cluster) con el objetivo de poder mostrar posteriormente un listado de los grupos que contiene cada cluster, ordenados por esta distancia.

Visualización

El propósito de esta parte del proyecto ha sido poder mostrar de manera clara y detallada el resultado conseguido en el resto de fases del proyecto.

Para ello se ha decidido crear una aplicación web que se comunica vía REST con el resto de implementaciones realizadas en el proyecto.

Como punto de partida se ha utilizado el siguiente proyecto open source

<https://github.com/akveo/blur-admin> sobre el que se han creado diversas pantallas que explotan la información recuperada, en muchas de ellas se ha hecho uso de los componentes de visualización ofrecidos por kibana y grafana, que permite exportarlos como iframes dentro de nuestra propia aplicación

A continuación se muestran algunas de las pantallas de la aplicación.

RealTime.

En esta página se muestra la evolución en tiempo real del flujo de entrada de respuestas. Número de respuestas recibidas en total y por segundo, cuántas respuestas son confirmaciones, gráficas y tablas con las respuestas agrupadas por país y ciudad



Figura-10. Página con información de respuestas recibidas en tiempo real.

Dashboards

En esta página se muestran diferentes dashboards sobre las tres fuentes de datos almacenadas en elasticsearch, mediante el uso de kibana

Dashboard de Respuestas. [Figura-11](#). Presenta diferentes gráficas con información de las respuestas recibidas según:

- País
- Ciudad
- Día de la semana
- Hora del día
- Categoría
- Localización. Visualización mediante mapa geoposicionado

Muestra también los eventos con mayor número de respuestas

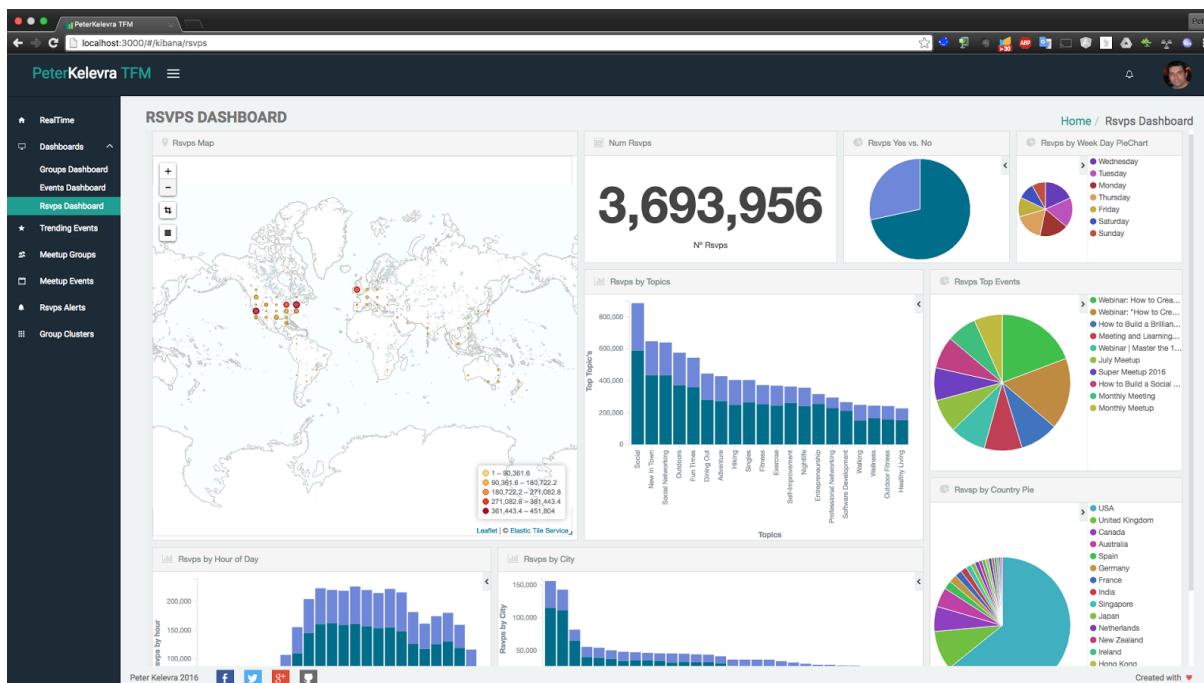


Figura-11. Página con información sobre respuestas capturadas

Dashboard de Grupos. [Figura-12](#). Presenta diferentes gráficas con información del número de grupos según:

- Fecha de creación
- País
- Ciudad
- Tamaño del grupo
- Categoría del grupo
- Localización. Visualizadas mediante mapa geoposicionado

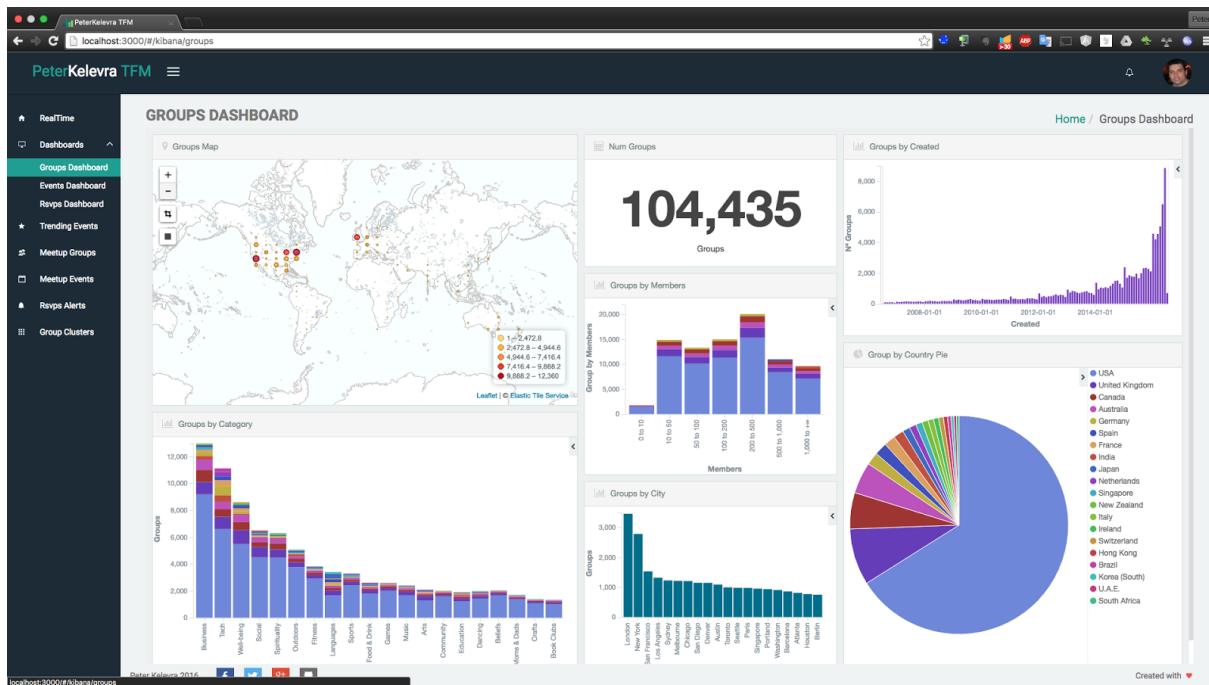


Figura-12. Página con información referente a Grupos

Dashboard de Eventos. [Figura-13.](#) Presenta diferentes gráficas con el número de eventos segun:

- País y Ciudad
- Día de la semana en la que se celebra el evento
- Hora del día en la que se celebra el evento
- Mes en el que se celebra el evento
- Nº de Confirmaciones recibidas
- Localización. Visualizadas mediante mapa geoposicionado

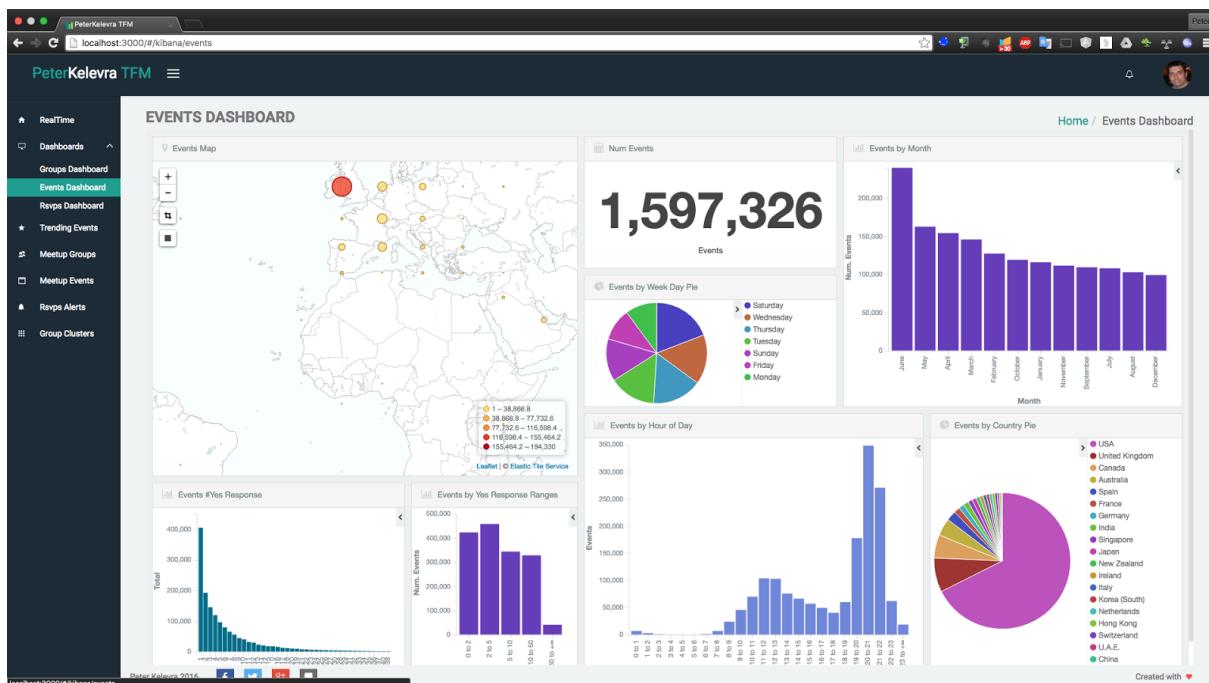


Figura-13. Página con información de eventos.

Listado de Grupos

Figura-14. En esta página se muestra un listado en el que se puede navegar sobre todos los grupos almacenados en elasticsearch, permitiendo aplicar filtros y ordenación de resultados. El objetivo con esta página al igual que la siguiente referente a eventos, ha sido mostrar la posibilidad de hacer páginas ad-hoc realizando consultas directamente sobre el api REST de elasticsearch

Figura-14. Página listado de grupos

Listado de Eventos

Figura-15. En esta página se muestra un listado en el que se puede navegar sobre todos los eventos almacenados en elasticsearch al igual que la página anterior permite filtrar y ordenar los resultados por ciertos campos.

Figura-15. Página listado de Eventos

Alertas

Como se ha comentado en la fase de análisis en tiempo real. Una de las tareas que se ha realizado, es la de notificar cuando se detectan eventos que están recibiendo una gran cantidad de respuestas. Como ya se indicó, estas alertas se enviaban a un topic kafka. Para hacer llegar estas notificaciones al usuario final a través de la aplicación web.

Para ello, se ha creado un servidor de aplicaciones con el framework play de scala. <https://www.playframework.com/>. Mediante el cual se ha implementado un servicio, encargado de consumir el topic kafka de notificaciones y enviarlas a un canal web socket. Como último paso del proceso, la aplicación web, implementada en angularjs, consume este web socket haciendo uso del modulo <https://github.com/AngularClass/angular-websocket>.

La notificaciones recibidas son mostradas finalmente al usuario mediante una ventana diálogo, la cual tiene un indicador con las alertas recibidas. Tal y como muestra la [Figura-16](#)

Las notificaciones recibidas pueden marcarse como leídas, para resetear el contador de alertas. No obstante, siempre podremos abrir la página de notificaciones que mostrar todas las recibidas desde que el usuario accedió a la aplicación, como se puede observar en la [Figura-17](#)

En esta misma página se puede configurar el umbral de respuestas recibidas por minuto por un grupo, para el cual queremos recibir notificaciones.

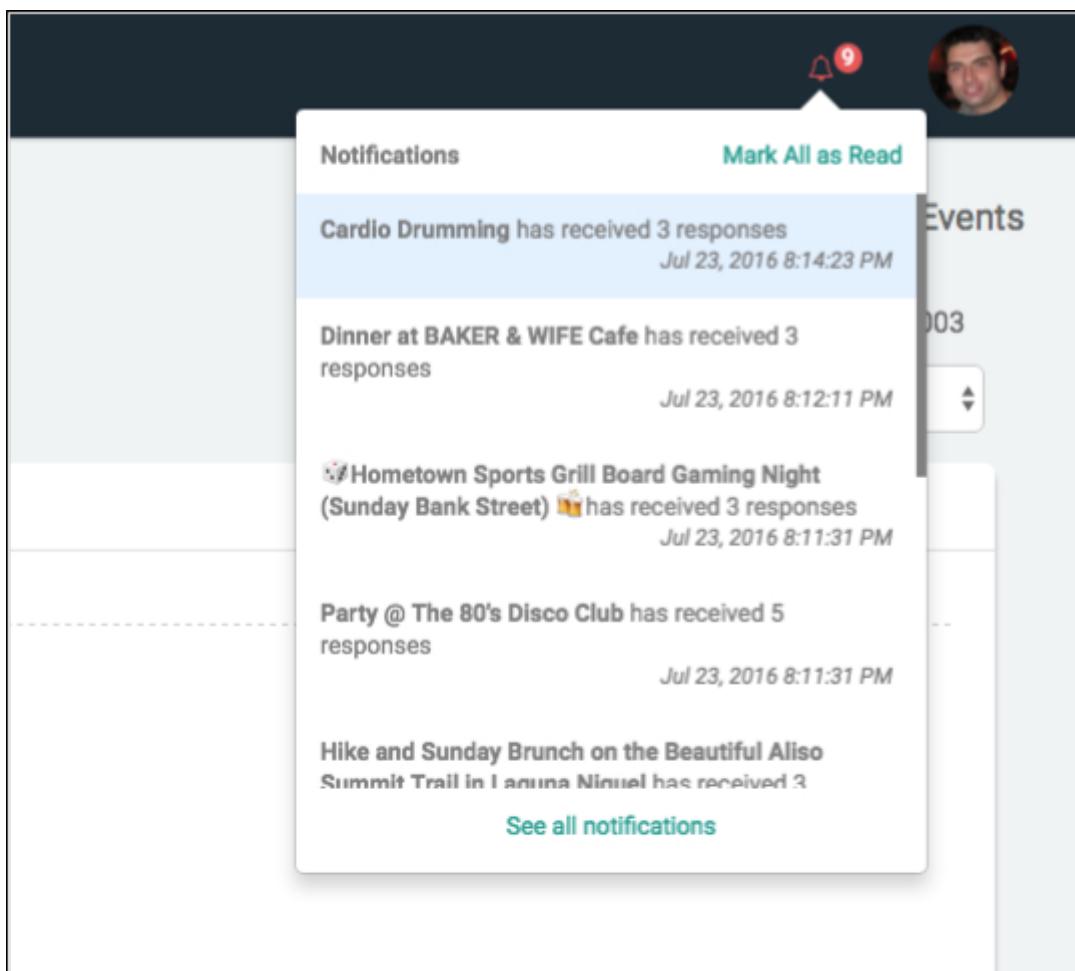


Figura-16. Sistema de notificación de alertas

Event	Event Date	Responses	Date
Saturday at Kingpin Suite Club - Live-Music-Night Feat. 5 different acts	Jul 23, 2016 8:30:00 PM	3	Jul 23, 2016 8:16:07 PM
Cardio Drumming	Jul 29, 2016 12:15:00 AM	3	Jul 23, 2016 8:14:23 PM
Dinner at BAKER & WIFE Cafe	Jul 24, 2016 1:00:00 AM	3	Jul 23, 2016 8:12:11 PM
Hometown Sports Grill Board Gaming Night (Sunday Bank Street)	Jul 25, 2016 12:30:00 AM	3	Jul 23, 2016 8:11:31 PM
Party @ The 80's Disco Club	Jul 25, 2016 1:00:00 AM	5	Jul 23, 2016 8:11:31 PM
Hike and Sunday Brunch on the Beautiful Aliso Summit Trail in Laguna Niguel	Jul 24, 2016 5:00:00 PM	3	Jul 23, 2016 8:10:55 PM
Meetup #3 - Migrating to React	Jul 28, 2016 8:00:00 PM	3	Jul 23, 2016 8:10:55 PM
Party @ The 80's Disco Club	Jul 25, 2016 1:00:00 AM	8	Jul 23, 2016 8:10:34 PM
New and Not so New Boomers Meetup	Aug 2, 2016 1:00:00 AM	3	Jul 23, 2016 8:10:34 PM
Mobile SEO Tips You Can't Afford To Miss	Sep 6, 2016 7:30:00 PM	4	Jul 23, 2016 8:10:34 PM

Figura-17. Listado de alertas recibidas

Trending Groups

Otro de los análisis en tiempo real llevado a cabo, ha sido la consolidación de Trending Events por país. Cuyo resultado es un listado de los diez eventos por país que han recibido mayor número de respuestas en la última hora.

Para mostrar dicha información al usuario final, realizamos un procedimiento similar al descrito en el punto anterior.

Un servicio se encarga de consumir los mensajes del topic kafka al cual se envían los trending groups obtenidos mediante Spark streaming. Estos mensajes una vez recibidos se envían a la aplicación web por medio de web sockets.

El resultado se muestra a través de una página web, como puede observarse en la [Figura-18](#), en la cual podemos ver el listado de eventos con el mayor número de respuestas recibidas en la última hora para cada país. Con la posibilidad de filtrar por nombre del mismo.

#	Event	Event Date	Resp.
1	Tech Lunch #4: React	Jul 27, 2016 12:00:00 PM	5
2	Saturday Swim at Torcy	Jul 30, 2016 9:00:00 AM	3
3	Walk /Promenade : nature, Parks, hidden places, discover... 16th district + Pub	Jul 24, 2016 1:00:00 PM	2
4	Boire un verre dans le 6ème arrondissement	Oct 6, 2016 7:30:00 PM	1
5	Friendly BBQ @ Picnic on a island in Paris* @ île saint Louis or Chalet des îles	Jul 24, 2016 2:00:00 PM	1
6	Our regular weekly Coffee Morning (Nice) - Distilleries Ideale - The Office!	Jul 24, 2016 10:30:00 AM	1
7	Picnic and music at Chateaux d'O	Jul 23, 2016 8:15:00 PM	1
8	Shut Up and Write! Get It Done! Weekday Edition	Jul 27, 2016 2:00:00 PM	1
9	Visite guidée nocturne du quartier Saint Germain des Prés	Jul 29, 2016 7:30:00 PM	1
10	* PUBSURFING ★ PUBCRAWL MADE IN PARIS	Jul 23, 2016 8:30:00 PM	1

#	Event	Event Date	Resp.
1	Games Part 2 - Sundays from 1pm	Jul 24, 2016 2:00:00 PM	18
2	Saturday at Kingpin Salle Club - Live-Music-Night Feat. 5 different acts	Jul 23, 2016 8:30:00 PM	11
3	Earlston Pub Crawl: Summer Edition	Jul 23, 2016 10:00:00 PM	6
4	Mobile SEO Tips You Can't Afford To Miss	Sep 6, 2016 7:30:00 PM	4
5	Pub Social - Drinks, Dancing & Karaoke @ The Miley, Rochford	Jul 23, 2016 9:30:00 PM	4
6	The Esk Adventures #2 - from Roslin to Lasswade & back	Jul 31, 2016 12:00:00 PM	4
7	City of London Treasure Hunt	Jul 24, 2016 3:00:00 PM	3
8	Drinks Night! #chinchin	Jul 23, 2016 9:00:00 PM	3
9	Evening Level 2 walk: whiteleaf cross, pulpit hill nature reserve and Chequers	Jul 28, 2016 9:00:00 PM	3
10	KEN'S OLYMPIC PARK WALK (24.07.16)	Jul 24, 2016 3:00:00 PM	3

#	Event	Event Date	Resp.
1	Como Maestro Pokemon quiero partir User Stories para tener impacto (CDMX)	Jul 28, 2016 1:30:00 AM	1
2	Language Exchange - Tandem Evening	Jul 26, 2016 2:00:00 AM	1
3	Los Ilusionistas 2	Jul 24, 2016 10:45:00 PM	1
4	Simple Sales Funnels : El arte de vender en internet.	Jul 24, 2016 3:00:00 PM	1

#	Event	Event Date	Resp.
1	HIKE VARGA -Fat & Calorie & Stress Rustenn-Kiara Hilli SI Inñarw-7AMI	Jul 24, 2016 1:00:00 AM	1

Figura-18. Visualización de Trending Events por País

Groups Clusters

En esta página, tal y como muestra la [Figura-19](#). Se muestra la información obtenida a partir del algoritmo de K-Means utilizado en la fase de análisis.

Además de poder ver la nube de etiquetas resultante del proceso, podemos ver los grupos que han sido clasificados dentro de cada cluster.

The screenshot shows a web application interface for managing group clusters. On the left, there's a sidebar with navigation links: RealTime, Dashboards, Trending Events, Meetup Groups, Meetup Events, Rave Alerts, and Group Clusters. The main area is titled "GROUP CLUSTERS". It displays two clusters: Cluster 1 and Cluster 2.

Cluster 1: Contains several word clouds representing different groups. One prominent cloud includes words like "culture", "wine", "arts", "cycling", "hiking", "class", "parent", "out", "photog", "book", "fun", "dog", "health", and "parent". Another cluster is labeled "Synergy National - Business Networking Group" with terms like "entrepreneurship", "smallbiz", "business strategy", "business-referral-networking", and "professional-networking".

Cluster 2: A modal window titled "Cluster 2 » Groups" lists 373 groups. Some examples include "SFLO-Entrepreneurs by Broward SCORE" (professional-networking, business-strategy, business-referral-networking, entrepreneurship, startup-businesses), "Southwest WI Business Networking" (smallbiz, business-referral-networking, professional-networking, entrepreneurship), "TEAM Valley Business Professionals" (business-referral-networking, professional-networking, business-strategy, entrepreneurship, business-entrepreneur-networking), and "East Cobb Business Network" (entrepreneurship, startup-businesses, professional-networking, business-entrepreneur-networking). Other groups listed include "Cornwall Startup Meetup", "Looking for a Co-Founder-MeetUp", and "Starved Rock Entrepreneurs".

The background of the application features large, semi-transparent word clouds for "working", "marketing", "ines", "fessional", "referral", "small", "development", "computer", "web", "softwaredev", and "technology".

Figura-19. Página listado de grupos pertenecientes a un cluster

Conclusiones

El objetivo del trabajo final de programa **Experto en Big Data** es realizar un proyecto en el que se pongan en práctica algunos de los conceptos y tecnologías aprendidos en el programa. Con tal propósito.

Se ha construido una arquitectura lambda, escalable horizontal y verticalmente, basada en un cluster **Hadoop / Yarn** sobre contenedores **Docker**. El cual, nos ha permitido, procesar, analizar y visualizar los datos de entrada provenientes de las respuestas a invitaciones de eventos de la plataforma Meetup.

Respecto a las tecnologías utilizadas, se han desarrollado diversas aplicaciones con el lenguaje de programación **Scala**, que hacen uso de **Apache Spark**, en el marco de **Sistemas Batch**.

Spark Streaming, en el marco de **Sistemas Real-Time / Near Real-Time**

Spark MLlib, en el marco de **Data Analytics and Machine Learning**. Específicamente se han probado diferentes algoritmos de analítica avanzada sobre los datos obtenidos, como son:

- Sistema de predicción de confirmación a invitaciones. Modelo de aprendizaje supervisado, planteado como un problema de regresión.
- Clasificación de Grupos en base a sus intereses. Modelo de aprendizaje no supervisado, mediante clustering con K-Means.

Como sistema de almacenamiento de datos, se ha utilizado **Elasticsearch**, servidor de búsqueda distribuido, basado en **Apache Lucene**, que trabaja con documentos en formato JSON, perfecto para nuestro caso de uso, que hace uso de los datos proporcionados por el **API REST de Meetup** que sirve documentos en este formato.

Con respecto al módulo de visualización. Se ha implementado una aplicación web donde se recogen todas las métricas y análisis llevados a cabo.

Esta aplicación, está basada en el framework **AngularJs** y en ella se han integrado, tanto soluciones de visualización de terceros sobre elasticsearch, como son **Kibana** y **Grafana**, así como implementaciones ad-hoc, mediante el uso de **Html**, **Javascript**, **Css** y diferentes librerías de visualización como **D3.js**

Para la integración entre los distintos módulos implementados se ha hecho uso de sistemas de mensajería distribuida como es el caso de **Apache Kafka**, sistema de caché distribuida **Redis**, así como la implementación de canales de comunicación mediante **WebSockets**.