Escuela Politécnica Superior

Desnuda tu alma de 140 en 140 caracteres

Grado en Ingeniería Informática

Trabajo Fin de Grado

Autor:
Teslem Sidi Hamudi

Tutor/es:
David Tomas Diaz

Universitat d'Alacant
Universidad de Alicante

Junio 2017

Para mi madre Lala, que como te prometí llegaría hasta aquí.

Para mi amigo Raul, gracias por tu apoyo incondicional.

Para todos los niños refugiados que no han tenido una educación y la suerte que he tenido yo.

Indice

1. Introducción 3
2. Objetivos 5
3. Metodología 6
   3.1. Análisis Funcional……6
   3.2. Extracción de información de Twitter……..7
   3.3. Tecnologías utilizadas……..7
   3.4. Herramientas para control y gestión software……..8
   3.5. Desarrollo e implementación……..11
4. Twitter como red social 13
5. Algoritmos de procesamiento de lenguaje natural 15
6. Arquitectura del sistema 18
   6.1. Capa de Persistencia 19
   6.2. Capa de Negocio: 22
      6.2.1. Modulo API 24
      6.2.2. Modulo Data 27
      6.2.3. Módulo Process Data 29
      6.2.3.2. Cálculo de TF-IDF 30
      6.2.3.3. Análsis de sentimientos: 30
      6.2.4. Modulo Starter 30
   6.3. Capa de Presentación 31
7. Casos de estudio 33
8. Conclusiones 41
9. Agradecimientos 42
10. Bibliografía 43

1. Introducción

Las redes sociales son el día a día, ya sea en el ámbito personal como en el ámbito profesional. Surgen como medios de comunicación para fomentar nuestra participación y cooperación, nos ayudan a relacionarnos con los demás de manera muy rápida y sencilla. En ellas se busca opinión, sugerencias, soluciones y sobre todo crear nuevos lazos sociales. Toda esta búsqueda se resuelve gracias aquello que publicamos y publican los demás, amigos, seguidores, empresas, etc.

Existen multitud de redes sociales tales como Facebook, Instagram, Whatsapp o Twitter, que surgen con el mismo fin pero aportando a sus usuarios funcionalidades diferentes. Twitter, por ejemplo, es una plataforma de comunicación con carácter de red social que limita sus mensajes a 140 caracteres. En 140 caracteres se puede expresar ideas instantáneas en poco tiempo, esto hace también que la lectura de los tweets no sea una tarea pesada.

En definitiva, la esencia de Twitter es la sencillez, pretende hacer llegar mensajes de manera veloz para que puedan ser leídos por nuestros seguidores y re-tuiteado hasta llegar a las masas.

Ahora bien, todas estas ventajas son a cambio de algo, nuestra privacidad. Nuestra privacidad está expuesta día a día a las grandes empresas que trabajan detrás de las redes sociales realizando acciones para conocer todo sobre nosotros, nuestros gustos, donde estamos y con quién estamos.

Con los perfiles sociales se abre la puerta de la intimidad a compañías que pueden incrementar sus ingresos gracias a nosotros, sus clientes y usuarios. A través de las redes sociales se venden preferencias y gustos a empresas que generan campañas de marketing específicas para cada usuario.

Las redes sociales se convierten bajo el consentimiento dado en “espías” que relatan todo sobre sus usuarios y muchas veces no son conscientes de ello. La información puede generar una gran ventaja competitiva para las empresas pero por otra parte también puede caer en manos de terceros que pueden utilizarla para otros fines.

En este proyecto se ha desarrollado una aplicación web para recoger toda la información que publicamos en Twitter. Es una herramienta capaz de analizar tweets, y extraer información útil para poder analizar rasgos personales de los usuarios.

Cada tweet por separado no aporta mucha información sobre un usuario, pero la agregación de todos ellos puede ayudar a identificar rasgos sobre él, temas sobre los que suele escribir, si son tweets positivos o negativos. Profundizando más en el estudio en un periodo de tiempo lo suficientemente largo podemos extraer estadísticas sobre su estado de ánimo.

Otras funcionalidades de la aplicación son la obtención de detalles como la geolocalización de cada usuario. Muchos tweets tienen especificado la localización exacta desde donde se han emitido. Esto permite saber zonas que suele frecuentar un usuario. También disponemos de la fecha y hora en la que se publicó cada tweet. Esto nos da la posibilidad de sacar estadísticas sobre las franjas horarias en las que suele twittear.

Este proyecto es una herramienta interesante desde el punto de vista empresarial para campañas de marketing. Las empresas pueden centrar sus esfuerzos en crear productos que se ajustan a la demanda de los usuarios. Una empresa con una herramienta de este tipo puede saber qué ofrecer y a quién ofrecerlo.

Este documento está dividido en las siguientes secciones: en primer lugar, se detallan los objetivos del proyecto y qué se pretende con él; a continuación, un apartado metodología donde se especifica todo el proceso de obtención de los tweets, procesamiento de la información obtenida y creación de una interfaz para mostrar los resultados; en el apartado arquitectura está desarrollada cada una de las tecnologías utilizas, la división en tres capas de la aplicación y el porqué de la elección de una arquitectura SOA; finalmente, los casos de estudio, donde se centrará en un conjunto de usuarios sobre el que se hace el estudio.

2. Objetivos

Hace pocos años éramos más reticentes a publicar sobre nuestra vida cotidiana, pero hoy en día nos hemos convertido en usuarios activos en ellas, esto es debido al crecimiento de la infraestructura de Internet. Con el crecimiento y evolución de Internet han crecido múltiples redes sociales como Twitter.

Twitter es una red social con carácter periodístico que impulsa la creatividad ya que el hecho de solo poder publicar 140 caracteres nos hace pensar y resumir aquello que queremos transmitir. Con un tweet nos hacen llegar mucha información al respecto de cualquier tema.

Twitter es dinámico y fácil de utilizar, sus usuarios generan millones de tweets por segundo. Cualquier tweet público puede ser comentado y juzgado, podemos expresarnos con total libertad, incluso hacia personas más mediáticas, ya sean políticos, famosos o seguidores.

En este proyecto se ha desarrollado una aplicación web, TweetMe, con el fin de aprovechar este flujo de información y convertir la información en datos que aportan significado para poder concienciar a nuestra sociedad sobre su privacidad.

El objetivo principal es poder analizar cualquier perfil público de Twitter a partir de su "screenName", el "screenName" identifica a cada usuario en Twitter, obteniendo un perfil social del usuario a partir del análisis de todos sus tweets. Este perfil obtenido incluye información de los temas que más aborda (a partir de los términos que utiliza con mayor frecuencia).

Otro aspecto interesante a estudiar en un perfil son los sentimientos que expresa en la red. TweetMe obtiene todos los tweets de un usuario y se determina si cada tweet publicado es de carácter positivo o negativo. El estudio de los sentimientos es un factor muy enriquecedor para la sociedad, puede ayudar a prevenir acoso escolar, marginación social entre otros.

La herramienta se centra principalmente en un usuario, pero también determina qué términos son más comunes en una zona, el usuario de zona con más términos, el usuario que más contenido genera y mayor número de seguidores generado, así como un ranking de usuarios. Estos datos pueden dar idea de aquellos usuarios más influyentes por zona.

En síntesis, este proyecto es una herramienta que extrae y almacena datos de Twitter, está soportado por una sólida arquitectura para el análisis de información utilizando técnicas de procesamiento del lenguaje natural para extraer información de los tweets. Toda esta información es mostrada al usuario a través de una interfaz web amigable.

3. Metodología

TweetMe es una herramienta para el análisis de los tweets publicados por los diferentes usuarios de Twitter. Es imprescindible disponer de un gran volumen de información que nos permita analizar y comparar los tweets de los diferentes usuarios. Para ello se ha decidido orientar TweetMe como un recolector de la información disponible en Twitter. Dicha información es, en un primer momento, almacenada para posteriormente ser filtrada, procesada y presentada al usuario de una forma que sea sencilla de interpretar.

En previsión de una futura expansión de la funcionalidad de TweetMe, toda la información que Twitter proporciona es almacenada, aunque de momento gran parte de dicha información se está descartando.

En este apartado vamos a detallar la metodología que hemos seguido para obtener el producto final. La metodología que hemos adoptado para construir el producto final se puede resumir en los puntos que se detallan a continuación.

3.1. Análisis Funcional

El análisis funcional es la primera fase del desarrollo Software, donde se especifican todos los requerimientos funcionales de un producto o aplicación. Sirve como contrato con el cliente, para explicar a los desarrolladores las funcionalidades a implementar, para estimar los esfuerzo requeridos para desarrollar el producto y para garantizar que tras la finalización del producto este se ajusta a las especificaciones del cliente.

Los requisitos funcionales de TweetMe se puede dividir en tres grandes grupos. Esta división viene sujeta por sus tres interfaces de usuario principales. La interfaz principal es una interfaz de zona, es decir, se centra en mostrar detalles de una zona geográfica específica, en ella se debe contemplar las siguientes funcionalidades:

- Una nube de Tags, o nube de palabras de mayor relevancia en una zona geográfica.
- Listar el usuario que más tweets genera.
- Listar el usuario que más seguidores tiene.
- Listar el usuario que más palabras relevantes tiene en esa nube de tags.
- Un mapa donde quedan reflejados los usuarios de zona según su última localización que viene dada por su último tweet.
- Un buscador para poder buscar cualquier usuario existente en Twitter y poder analizarlo.

La segunda interfaz es producto de la búsqueda de un usuario específico. Se resume en las siguientes funcionalidades:

- Perfil del usuario: nombre, nickname, cantidad de seguidores, cantidad de tweets.
- Mapa donde queda geolocalizado según su último tweet.
- Términos más relevantes expresados en una nube de tags.
- Una línea temporal de los tweets de un usuario y si son de carácter positivo o negativo.
- Horas más frecuentes que suele escribir tweets.

La tercera interfaz es la interfaz de login. Debe ser un formulario que permite a los consumidores de la aplicación acceder a través unas credenciales.

TweetMe está enfocada como herramienta de procesamiento de tweets aplicando un algoritmo de lenguaje natural para determinar términos más relevantes en una colección de tweets. La aplicación debe ser capaz de aplicar dicho algoritmo para determinar esos términos de mayor importancia y generar un componente visual para mostrarlo en la plataforma.

Los tweets se deberían categorizar según si son mensajes positivos o negativos. Esto aplicado a múltiples tweets de un usuario nos ayuda a determinar si es una persona con mayor tendencia positiva o negativa.

La aplicación debe permitir registro y autentificación para que los usuarios puedan consultar todos los detalles de una zona o hacer una búsqueda de usuarios. Un usuario que inicia sesión interactúa con la aplicación y realiza una búsqueda de perfiles de Twitter. La búsqueda se puede realizar introduciendo el "nickname" de ese perfil o haciendo doble click sobre algún usuario de zona. Los usuarios de zona aparecen listados en la primera interfaz.

Esta búsqueda de un usuario hace un recolección de todos sus tweets y un análisis de los mismos comparándolos con los tweets zona.

De los usuarios buscados debemos mostrar la información básica de perfil, una línea temporal con todos los tweets y una representación gráfica que determine a grandes rasgos si los tweets de un usuario son positivos o negativos.

Finalmente, se debe mostrar un mapa para ubicar al usuario sobre el que se realiza la búsqueda. Ahora mismo el estudio solo se realiza a una ciudad, Alicante, pero en un futuro se puede trasladar a cualquier ciudad.

3.2. Extracción de información de Twitter

En primer lugar, habiendo ya especificado los requisitos funcionales de la aplicación, tenemos que saber qué información nos ofrece el API de Twitter y si es posible satisfacer nuestros requisitos.

Twitter ofrece una excelente documentación sobre su API para que los desarrolladores puedan interactuar con la misma sin dificultad.

Si nos dirigimos a la siguiente URL https://dev.twitter.com/rest/public podemos encontrar toda la información necesaria sobre el API, cómo conectarnos a él, todos los servicios que ofrece y los resultados de las diferentes consultas a los métodos del API; hay diferentes maneras para conectarse al API que detallaremos más adelante.

En esta fase vimos que el API nos brinda toda la información que necesitamos para construir TweetMe. En algunos casos, obtener los datos necesarios requiere hacer múltiples consultas, ahora bien, ¿cómo conseguimos estos datos y dónde los almacenamos?

3.3. Tecnologías utilizadas

Teniendo en cuenta que el API que ofrece Twitter es un API REST y que el objetivo es crear una arquitectura distribuida no ligada a las tecnologías utilizadas, se definió la arquitectura del proyecto como una arquitectura basada en SOA y con servicios REST. La elección de la arquitectura de un proyecto no es tarea sencilla y requiere de experiencia o al menos un trabajo de búsqueda para saber qué tecnologías son afines y cumplen nuestras expectativas.

En este proyecto se eligió una arquitectura SOA gracias a las ventajas que nos ofrece, más adelante señaladas, y servicios REST que implementan la misma. Determinar la arquitectura como un conjunto de módulos independientes basados en arquitectura SOA es un gran avance, pero seguimos sin tener nuestros datos y sin saber cómo consultar el API.

Se hizo una búsqueda de diferentes librerías que implementan muchas operaciones del API de Twitter y que tan solo había que importarlas, pero en esta búsqueda encontramos algo mucho mejor. Desde primer momento se eligió el framework Spring Boot como herramienta para construir la lógica de negocio, pero además este framework ofrece un módulo exclusivo para implementar todas las operaciones del API de Twitter, además de otros módulos para conectar con otras redes sociales como Facebook o Instagram.

Spring Boot[1] implementa todas las operaciones y esto facilita el trabajo muchísimo, ya que Spring también ofrece una gran documentación al respecto y para nosotros es transparente la conexión que él hace al API, no nos tenemos que preocupar de ello, Spring ya lo hace por nosotros.

Se determinó como lenguaje para la capa de negocio Java apoyado sobre el framework Spring Boot; Spring Boot es un framework muy utilizado a día de hoy en las aplicaciones web, es de software libre y mantenido por múltiples desarrolladores.

Para el almacenamiento de todos los tweets recuperados del API hemos utilizado una base de datos no relacional, MongoDB[2]. En un primer momento todos los tweets recuperados se almacenan aquí para tener una base de datos con los tweets sin procesar y poder recuperarlos en cualquier momento.

Ahora bien, una vez aplicamos el algoritmo del Lenguaje Natural sobre los tweets, los datos resultantes los almacenamos en una base relacional, utilizando para ello el gestor de bases de datos PostgreSQL[3]. Esta base de datos es un conjunto de tablas resultado que almacenan los datos finales tras aplicar el procesamiento de datos.

Para aportar más valor a la aplicación se ha utilizado AngularJS[4], desarrollado por Google. Es un framework para la capa de presentación basado en el concepto de SPA, **Single Page Application**. AngularJS se combina de manera excelente con Spring ya que consume todos los servicios REST generados en la capa de negocios.

3.4. Herramientas para gestión y control software

Llegados este punto se debían fijar ya nuestras tareas para llevar a cabo todo el proceso de construcción de TweetMe. Se han elegido herramientas tales como Trello[5], como panel que nos permite visualizar el estado de nuestras tareas en cada instante.

En todo proyecto software es fundamental el análisis, la estimación de esfuerzos y sobre todo la planificación. Estas tareas junto con múltiples más constituyen el ciclo de desarrollo software y cada una de ella aporta significado en él.

[1] https://projects.spring.io/spring-boot/
[2] https://www.mongodb.com/es
[3] https://www.postgresql.org/
[4] https://angularjs.org/
[5] https://trello.com/

Una vez tenemos claro nuestros requisitos funcionales, arquitectura y tecnologías a utilizar, hay que llevar una gestión del tiempo de las tareas, de manera que todos los componentes del equipo tengan una visión en todo momento de la evolución del proyecto. Trello proporciona esta funcionalidad. Al principio las tareas son más genéricas y conforme se ha ido avanzando en el proyecto las tareas han sido más concretas.

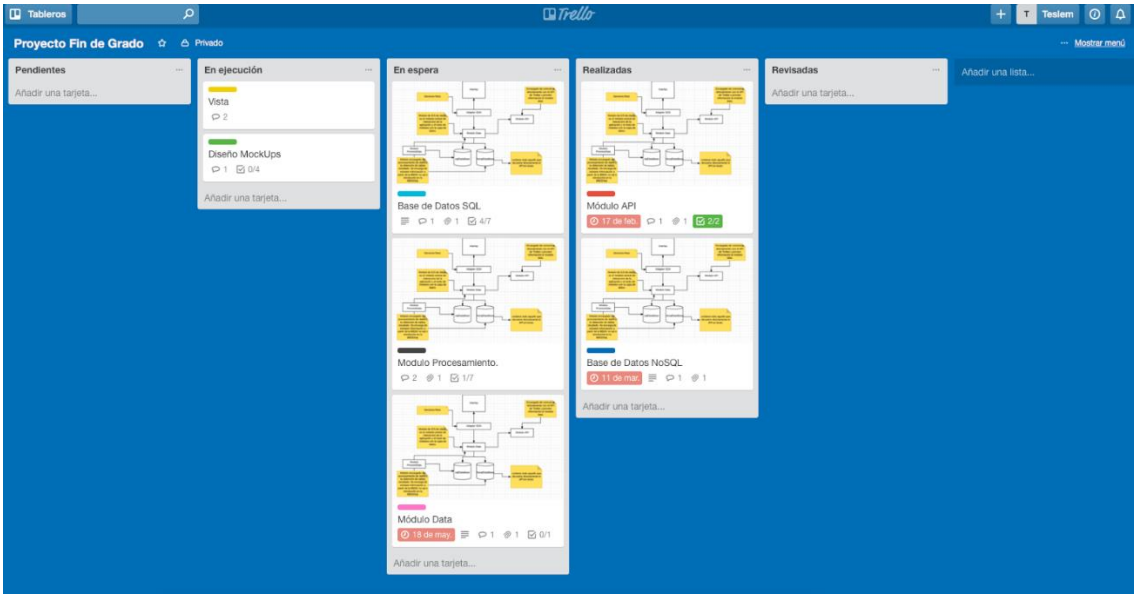


Figura 1 – Captura de página principal de Trello

Como se puede observar en la captura anterior, el escritorio de Trello se constituye de múltiples paneles. Cada uno de ellos indica la fase de las tareas. Esta herramienta nos permite asignar a cada tarea una fecha de finalización y un checklist para ir completando la tarea. Todo esto hace de Trello una aplicación que ofrece mucho en su versión gratuita y muestra el estado del proyecto a simple vista.

Las tareas se han dividido principalmente por los grandes bloques de desarrollo asociados a cada módulo que se detallan en esta memoria más abajo. En cada tarea se han ido añadiendo comentarios del progreso de la misma o links de importancia para su futuro uso o anexo a la bibliografía.

En este ciclo de desarrollo no se ha pretendido crear una metodología ágil basada en Scrum[6] ya que no hay múltiples desarrolladores, pero en previsión de un crecimiento de la misma aplicación se pueden utilizar herramienta de gestión de proyectos mucho más potentes como pueden ser Jira[7]. Jira nos permite el mismo control pero de manera mucho más sofisticada, tiene enlaces directos con el gestor de versiones de código SourceTree[8].

Teniendo una primera versión de todas las tareas que hay por desarrollar y el tiempo estimado para ello solo falta ponerse manos a la obra. Esta fase del desarrollo suele ser muy costosa y con múltiples errores de configuración de archivos. En primer lugar tenemos que asegurarnos de tener siempre una copia de aquello que estamos desarrollando, para esto se ha utilizado Git y Bitbucket[9].

[6] https://proyectosagiles.org/que-es-scrum/
[7] https://es.atlassian.com/software/jira
[8] https://www.sourcetreeapp.com/
[9] https://bitbucket.org/product

Git es una excelente herramienta de control de versiones. Permite visualizar quién ha realizado los cambios, las fechas, versiones anteriores con su respectivo commit y sobre todo tener protegido nuestro código. Con Git nos aseguramos que ante cualquier fallo podemos recuperar nuestros datos anteriores con solo revertir los cambios.

En proyectos grandes se asocia a la herramienta de control del proyecto como Jira y permite una alta gestión del mismo.

En la Imagen 2 se observa todos los commits de la rama develop, persona responsable del commit y la fecha del mismo. Cada commit corresponde a una versión del proyecto a la que podemos volver en cualquier momento.

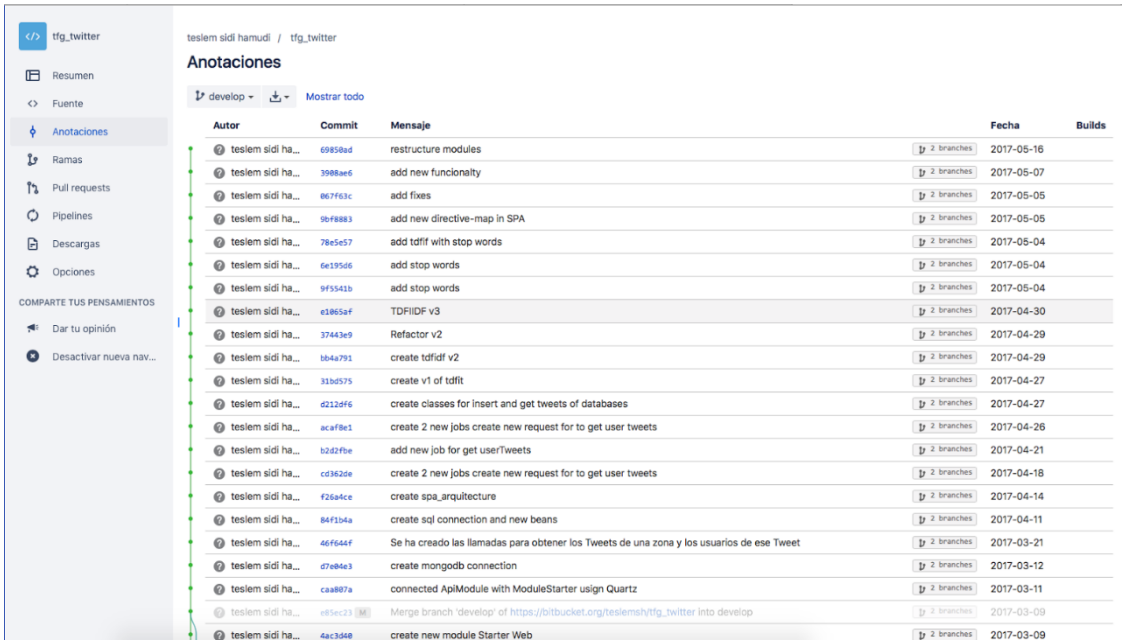


Figura 2 – Captura de página SourceTree.

Cabe destacar que cuando hay múltiples desarrolladores es bueno crear diferentes ramas de desarrollo para evitar conflictos en la misma rama y tener una rama final de la que parten todos ellos.

Para poder construir nuestros diagramas UML o bien para crear representaciones gráficas de las interacciones de los componentes de la aplicación se ha utilizado la herramienta Draw.io[10], nos permite elaborar diagramas online para evitar tener que descargar Software de escritorio.

Draw.io posee una interfaz muy sencilla de utilizar, como podemos observar la imagen posee un panel lateral a la izquierda donde podemos elegir cualquier tipo de diagrama ya sea UML, Entidad Relación, de negocio, etc...

Cuando elegimos cualquiera de estos diagramas la herramienta nos ofrece diseños predeterminados como base para nuestros proyectos, estos diseños los podemos diseñar y modelar a nuestro gusto, además de permitir incluir imágenes externas que se obtienen a través de una búsqueda en Google. La Imagen 3 muestra un ejemplo de los diagramas que se pueden generar con Draw.io.

[10] https://www.draw.io/

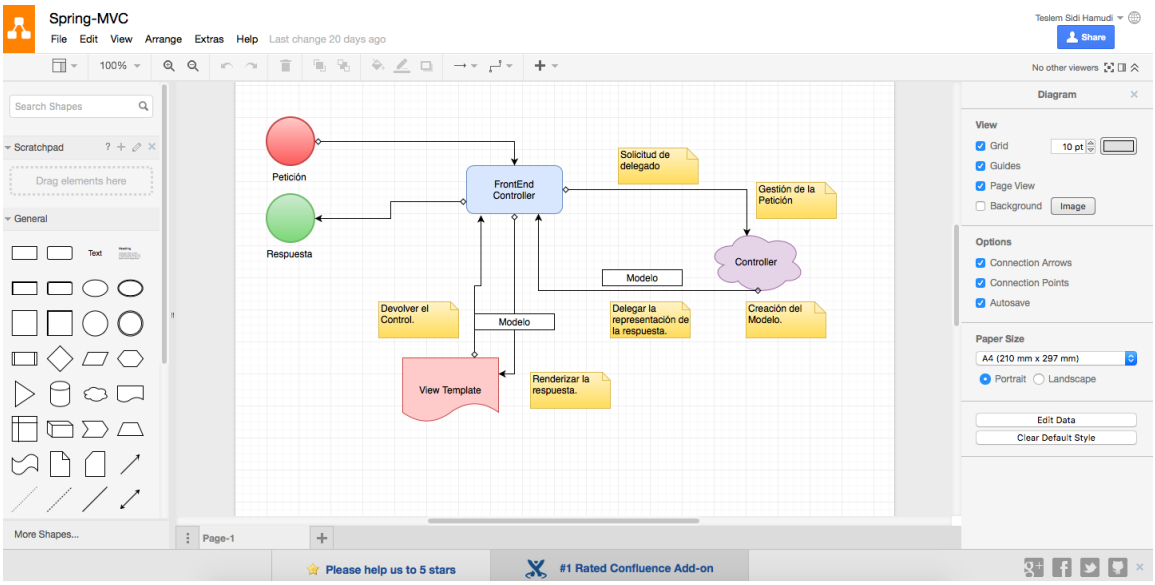


Figura 3 – Captura de Draw.io

El registro en esta aplicación es totalmente gratuito. En este proyecto se ha asociado a una cuenta de Google para poder tener guardados todos los proyectos y poder acceder a ellas desde cualquier lugar ya que funciona igual que Google Docs.

3.5. Desarrollo e implementación.

Al finalizar la planificación inicial, se puso en marcha la recogida de datos. En un primer momento tan solo obteníamos los datos en formato JSON pero no los almacenamos. En este punto llega la capa de persistencia. La capa de persistencia o almacenamiento de datos es una de las capas de mayor importancia en una aplicación, un buen diseño de la misma agiliza el desarrollo del resto de capas.

Puede parecer que se ha desarrollado la capa de negocio antes que la capa de persistencia, pero esto no fue así. Para poder construir un primer modelo lógico de nuestra base de datos relacional teníamos que tener algunos tweets para saber su estructura y cómo vamos almacenar los datos.

Todos estos tweets fueron almacenados en una base de datos noSql para su posterior procesamiento. Estos tweets nos aportan una idea de todas las entidades que queremos construir en nuestro modelo relacional.

El modelo relacional no fue creado desde un principio de manera íntegra, si no de manera iterativa acorde con las vistas. Es decir, como tenemos todos los datos en una base de datos noSql no nos preocupa de alguna manera hacer una base de datos relacional que almacena todo desde un principio, si no haciendo pequeñas funcionalidades. Por ejemplo, si queríamos listar usuarios en la vista, creamos el mockup de esa pantalla y las operaciones en la capa de negocio que nos devuelven usuarios. Entonces, a partir de los datos almacenados en la base de datos noSQL se creó una nueva entidad en nuestro diagrama Entidad Relación.

Esta forma de trabajar por ciclos permite ver cómo la aplicación se va construyendo como un puzzle y ensamblando poco a poco. Esta manera de trabajar se asemeja a los llamados *sprints* en una metodología ágil como Scrum. Es una forma de ir haciendo todas las semanas un poco de todo y no perder de vista ninguna capa, sobre todo no crear operaciones que puedan ser deshechas en un futuro.

En resumen, la elección de una metodología de trabajo no es una tarea sencilla, conlleva pensar, experiencia y compromiso. Claro está que el desarrollo software está sujeto a las previsiones, pero estas a veces cambian obligándonos a aumentar o dilatar las fechas que teníamos previstas en un principio.

En los inicios del desarrollo software existían técnicas muy tradicionales y sujetas al proceso, sin tener en cuenta el factor humano ni el producto final. Hoy en día hay una gran apuesta por las metodologías ágiles que enfocan sus esfuerzos en el individuo y en la colaboración de todos los componentes del equipo.

4. Twitter como red social

En las últimas décadas y con el lanzamiento de la llamada **Web 2.0** renace con ella una nueva manera de comunicarnos y relacionarnos. Esta nueva era de Internet permite a grandes empresas aprovechar el fenómeno de internet como medio para desplegar aplicaciones que conecten las masas, esto permite que las masas colaboren y aporten su opinión, es una nueva manera de escuchar y transmitir las opiniones.

Todo este nuevo flujo de información tiene como canal las llamadas Redes Sociales, la definición de este concepto viene de la mano de **Andreas M. Kaplan y Michael Haenlein.** Según ellos las redes sociales o Social Media son "un grupo de aplicaciones disponibles en Internet, construidas y basadas tecnológica e ideológicamente en la Web 2.0 que permiten la creación y el intercambio de contenido generado por el usuario".

Twitter es un servicio de microblogging creado en California, Estados Unidos, por Jack Dorsey, Evan Williams, Biz Stone y Noah Glass en 2006. Desde el año de su lanzamiento Twitter ha sido un éxito rotundo.

Twitter es una herramienta que nos permite relacionales de manera bidireccional con aquellos que nosotros elegimos. Es una plataforma que restringe cada publicación a 140 caracteres. Esta limitación es lo que hace especial a Twitter y causa controversia ya que para algunos es un defecto mientras que para otros impulsa la imaginación.

Esta relación bidireccional se puede resumir en:

- Followers: en esta lista están agrupados aquellos usuarios que deciden seguirnos.
- Friends: en este segmento se localizan usuarios que nosotros seguimos.

Twitter nos permite seguir todos aquellos usuarios que deseemos. Solo es necesario solicitar permiso para aquellos usuarios que tenga configurado su perfil como privado, pero esto no suele ser lo habitual ya que una de las características de Twitter es el hecho de que casi todos los perfiles son públicos.

El conjunto de usuarios que seguimos se convierte en nuestro timeline, o lo que es lo mismo, la lista de últimos tweets generados por el segmento de friends. La calidad de un timeline es nuestra responsabilidad, nosotros decidimos a quién seguir y por tanto decidimos aquellos que vemos en nuestro timeline.

A Continuación mostramos algunas estadísticas interesantes sobre Twitter recogidos por Kit Smith [11]en el blog **brandwatch** sobre las estadísticas de los usuarios:

- Existen **310 millones** de usuarios activos al mes.
- Se han creado un total de **1,3 mil millones** de cuentas.
- De estas, un **44%** crearon una cuenta y la cerraron antes de mandar un tuit.
- Sólamente **550 millones** de personas han mandado un tuit alguna vez.
- **500 millones** de personas visitan el sitio al mes sin iniciar sesión.
- Un **29,2%** de los usuarios de redes sociales en Estados Unidos son usuarios de Twitter.
- El 80% de los usuarios activos acceden a la plataforma a través del móvil.
- El promedio de seguidores es **208**.
- **391 millones** de cuentas no tienen ni un seguidor.
- Katy Perry tiene el mayor número de seguidores, con más de **87 millones.**

[11] https://www.brandwatch.com/es/2016/06/44-estadisticas-twitter-2016/

- Los periodistas conforman el **24,6%** de las cuentas verificadas.
- El **83%** de los líderes mundiales están en Twitter.
- Un **79%** de la cuentas se encuentran fuera de Estados Unidos.Twitter estima que **23 millones** de sus usuarios son *bots*.

En definitiva, Twitter es un portal de información y opinión pública que genera gran cantidad de información por segundo y que tiene gran peso en la sociedad más fan de esta red social.

En Twitter hay perfiles públicos y privados, como desarrolladores podemos consumir toda la información de aquellos perfiles públicos a través de sus múltiples complementos y widgets pensados para desarrolladores.

En su página oficial https://dev.twitter.com/docs vienen detallados todos los productos y servicios que ofrece. Podemos destacar los siguientes productos:

- Twitter for Websites: es un conjunto de utilidades que permiten incluir botones, widgets y scripts en cualquier sitio web permitiendo la interacción con Twitter.
- Kit of Twitter: es un SDK para aplicaciones Android, iOS. Permite añadir contenido dinámico en aplicaciones móviles para hacerlas mucho más atractivas.
- Cards: Es una herramienta que permite añadir información adicional a los links, fotos o videos cuando son compartidos vía Twitter.
- OAuth: Permite a los usuarios conectarse a Twitter de forma segura.
- REST API : permite el acceso a través de peticiones de tipo REST a toda la funcionalidad de Twitter.
- Streaming API: ofrece la posibilidad de realizar llamadas a REST API de un largo periodo de tiempo permitiendo así sincronizar en tiempo real las actualizaciones hechas por los usuarios.
- Ads API: permite integrar anuncios a determinados clientes de Twitter para que puedan crear campañas de publicidad.

5. Algoritmos de Procesamiento del Lenguaje Natural

Para poder extraer información del lenguaje natural que encontramos en los tweets, se ha decidido utilizar el algoritmo TF-IDF para calcular las palabras más relevantes de cada usuario.

Existen diversas implementaciones disponibles en forma de librerías de software libre. En TweetMe se ha optado por utilizar Apache Lucene[12]. Esta librería fue creada en el año 1999 y desde entonces ha ido evolucionando y adaptándose a las necesidades que los usuarios han ido demandando. Esta librería goza de mucha popularidad en la industria del software. Son muchas las empresas que han decidido incluir Lucene en su software. El principal uso que se le suele dar a Lucene es para indexar documentos y permitir así una búsqueda a partir de un término dado. Lo que nos interesa de Lucene no es su capacidad para indexar documentos, sino la implementación del algoritmo de TF-IDF que incluye para poder realizar dicha indexación.

Para el análisis de la información recolectada, TweetMe hace uso del algoritmo TF-IDF. Este algoritmo nos proporciona información sobre cómo de relevante es una palabra en un documento dentro de una colección o corpus. TF-IDF es el producto de dos variables estadísticas: la frecuencia de un término (Term Frequency) y la inversa de la frecuencia de un documento (Inverse Document Frequency).

- TF:

La forma más simple de calcular la frecuencia de un término en un documento es contar las veces que dicho término aparece en el documento. No obstante, este método no es el más aconsejable ya que no se está teniendo en cuenta la longitud del documento. En un documento extenso, un término aparecerá repetido muchas más veces que en uno más corto, aunque proporcionalmente puede que ambos términos sean utilizados con la misma frecuencia. Es por ello que se tiende a usar una fórmula más compleja la cual normaliza los resultados para que puedan ser comparables. De esta forma, la fórmula quedaría de la siguiente manera:

$$tf(t,d) = 0.5 + 0.5 \cdot \frac{f_{t,d}}{max\{f'_{t',d} : t' \in d\}}$$

Donde:

$f_{t,d}$ = el número de veces que aparece el término en ese documento.

$max\{f'_{t',d} : t' \in d\}$ = el máximo de veces que un término aparece en el documento.

- IDF:

La inversa de la frecuencia de un documento nos dice cuánta información aporta un término en el conjunto de todos los documentos. Para su cálculo una posible fórmula es:

$$idf(t,D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

[12] https://lucene.apache.org/core/

Donde:

N = es el número de documentos en la muestra o corpus.

$|\{d \in D : t \in d\}|$ = es el número de documentos en los que al término t aparece.

Así pues, TF-IDF es igual a:

$$tfidf(t, d, D) = tf(t, d) \cdot idf(t, D)$$

El siguiente ejemplo nos muestra cómo utilizar el algoritmo:

Disponemos de dos documentos los cuales contienen términos cuya frecuencia de aparición es:

Término	Documento 1	Documento 2
Biblioteca	1	1
Archivo	1	1
Documento	2	2
Museo	0	3
Libro	1	0

Vamos el TF del término *Biblioteca*:

Tf("Biblioteca",Documento 1) = 0.5 + 0.5 * 1 / 2= 0.75

Tf("Biblioteca",Documento 2) = 0.5 + 0.5 * 1 / 3 = 0.67

Lo que se desprende de los resultados es que aunque en ambos casos el termino se utiliza una vez en cada documento, en el caso del documento dos su frecuencia es menor porque hay términos que son más frecuentes .

Calculemos ahora el IDF:

IDF("Biblioteca",Documento 1) = log (2/2) = 0

IDF("Biblioteca",Documento 2) = log (2/2) = 0

Como el termino *Biblioteca* aparece en ambos documentos el resultado del cálculo del IDF es igual a cero.

Por último calculemos el TF-IDF:

TFIDF("Biblioteca",Documento 1) = 0.6 * 0 = 0

TFIDF("Biblioteca",Documento 2) = 0.5714 * 0 = 0

Como el resultado es igual a 0 deducimos que ese término no es relevante ya que se utiliza en ambos documentos y por tanto no es significativo de ningún documento.

En el caso de TweetMe, un documento es el conjunto de todos los tweets de un usuario. Como el API de Twitter nos proporciona los tweets de manera individual, es necesario concatenar todos los textos de los tweets para generar un documento. Durante este proceso se ha procedido a eliminar palabras comunes como preposiciones, artículos, etc; para evitar que los resultados

se vean alterados. Un listado de todas estas palabras vacías ha sido almacenado en la base de datos. El proceso para eliminar estas palabras poco relevantes se compone de los siguientes pasos:

1) Tokenizar el texto de cada tweet o lo que es lo mismo, generar un listado de todas las palabras que componen el tweet.
2) Comprobar que palabras del listado pertenecen al conjunto de palabras vacías almacenado y proceder a su eliminación.
3) Añadir al documento todas aquellas palabras relevantes que no han sin eliminadas en el apartado anterior.

Aunque existen otros procedimientos para realizar este filtrado de palabras, se ha optado por este método porque aporta una mayor flexibilidad. El hecho de tener el listado de palabras vacías almacenado en la base de datos nos permite añadir fácilmente cualquier palabra que consideremos irrelevante. Hay que tener en cuenta que el lenguaje utilizado por muchos usuarios de Twitter es un muy coloquial y lleno de abreviaturas que cambian según las "modas" del momento.

Disponer de una colección de documentos lo suficientemente grande es algo esencial para que los resultados sean correctos. Por este motivo TweetMe realiza una búsqueda por zona, de forma que a partir de una población dada se obtienen todos los tweets publicados por cualquier usuario. Nos centramos en una población determinada por acotar los usuarios, ya que en Twitter hay cientos de millones de tweets. Acotar geográficamente nos puede permitir en un futuro comparar diferentes zonas y analizar si la zona afecta al tipo de perfil.

Esta búsqueda por zona se realiza de forma periódica con el fin de ampliar y mantener actualizado el corpus. Adicionalmente, se pueden buscar y agregar los tweets de un usuario en concreto para poder así realizar un análisis detallado de ese usuario en particular.

Además de calcular las palabras más relevantes, TweetMe realiza un análisis de los sentimientos de cada tweet. El resultado de este análisis nos dice si ese tweet es positivo o negativo. Si disponemos de una cantidad lo suficientemente grande de tweets, este análisis nos permite sacar estadísticas de cada usuario y determinar el estado de ánimo del usuario en determinados momentos.

Para realizar analizar los sentimientos de un tweet se ha hecho uso de una herramienta [13] desarrollada por el Grupo de Procesamiento Del Lenguaje Natural y Sistemas de Información (GPLSI[14]) de la Universidad de Alicante. Esta herramienta consiste en un API REST que está disponible a través de internet para todos aquellos usuarios que dispongan de un token que les autorice a utilizarlo.

El API dispone de dos servicios que permiten realizar el análisis de los sentimientos para textos en español y para textos en inglés. Estos servicios aceptan como entrada un JSON en el que se le envía el texto de cada tweet. La respuesta de este servicio es un documento en formato JSON en el que nos dice si el texto enviado es positivo o negativo y la intensidad con la que este sentimiento se asocia al texto.

[13] http://gplsi.dlsi.ua.es/services/socialobserver3/doc/#!/sentimentanalysis/analyseES
[14] https://gplsi.dlsi.ua.es/gplsi13/es

6. Arquitectura del sistema

TweetMe está dividida en múltiples módulos que actúan de manera independiente y a la vez están comunicados. De esta manera conseguimos un bajo acoplamiento y una alta cohesión. Esto se consigue adoptando una arquitectura SOA.

SOA es un modelo de arquitectura basado en el paradigma orientado a servicios, las aplicaciones que se basan en este tipo de arquitectura se descomponen en pequeños componentes o funcionalidades, cada componente es un servicio.

En TweetMe se ha querido reflejar esta arquitectura ya que aporta flexibilidad en cuanto a añadir nuevas funcionalidades en el futuro, respetando así la integridad de la aplicación y consiguiendo un bajo acoplamiento.

Ahora bien, para poder llevar a cabo esta arquitectura, se hace necesaria la utilización de algún estándar de diseño de servicios web. En nuestro caso hemos optado por servicios de tipo REST. Cabe destacar que una arquitectura SOA no está sujeta a la utilización de servicios Web, pero los Servicios Web si son la manera más fácil de implementar una arquitectura SOA.

Los servicios REST son un intermediario entre dos aplicaciones, Cliente y Servidor, comunicados a través del protocolo HTTP con el objetivo de obtener datos en formato XML o JSON.

Como alternativa a los servicios REST, se pueden utilizar servicios SOAP, pero debido a la complejidad de este tipo de servicios se ha optado por utilizar REST, ya que utilizando transferencia de datos en formato JSON se convierten en un complemento idóneo para completar una interfaz AngularJS.

Los servicios REST siguen las siguientes premisas:

- **Cliente/Servidor:** los servicios web definen un intermediario de comunicación entre un cliente que realiza una petición a un proveedor o servidor.
- **Stateless:** los servicios de tipo REST no mantiene estado. Cada petición que hacemos a un servicio es independiente de la anterior, es decir, el resultado de una no es parámetro de otra que la precede. REST no mantiene estado del cliente.
- **Arquitectura en capas:** una de las grandes ventajas de REST es la escalabilidad que ofrece. Define una arquitectura de capas independiente donde el cliente tan solo se comunica con el servicio sin necesidad de saber dónde está el servidor.
- **Servicios Uniformes:** REST utiliza los métodos HTTP básicos (POST, GET, DELETE y PUT).
- **Recursos:** cada servicio tiene asociada un URI para poder conectar con él.

En la Imagen 3 se muestra un diagrama que nos ayuda a detallar la división de estos módulos y su interacción con la aplicación.

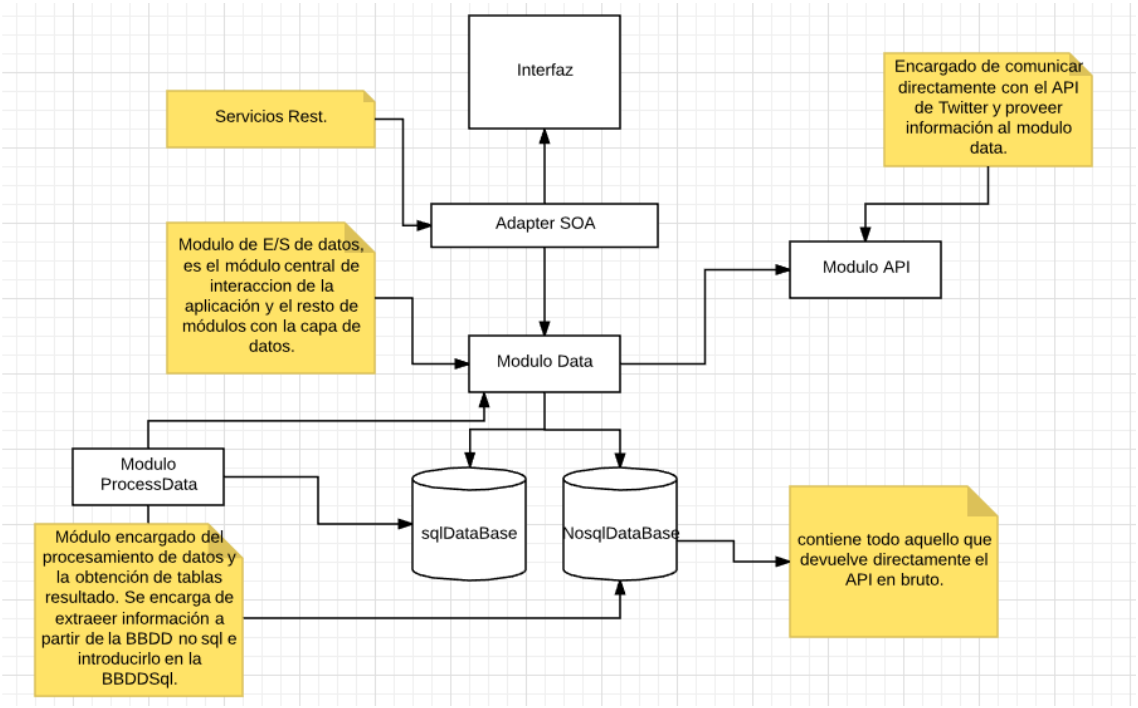


Figura 4 – Esquema de la arquitectura de TweetMe

Como podemos ver, la aplicación se ha distribuido por módulos. Cada uno de ellos es un artefacto que se despliega y empaqueta para poder ser consumido por los demás.

Si distribuimos este diagrama en tres entidades genéricas obtenemos la típica división, FrontEnd, BackEnd y BBDD lo que representa una arquitectura de tres capas. Disponemos entonces de una capa de datos para almacenar todos los datos que gestiona nuestra aplicación, usuarios, tweets y datos intermedios, una capa intermedia que se encarga de procesarlos y una capa para presentarlos.

Una división de tres capas se suele resumir en:

- **Capa de Persistencia:** esta capa se encarga del almacenamiento de datos. En este caso lo que se ha utilizado son dos bases de datos, una no relacional, MongoDB, y una relacional, PostgreSQL.
- **Capa de Negocio:** es el corazón de nuestra aplicación, es la capa que se encarga de la gestión de la lógica de la aplicación.
- **Capa de Presentación:** es la capa que interacciona directamente con el usuario y ofrece una representación más amigable de los datos resultantes de todas las operaciones.

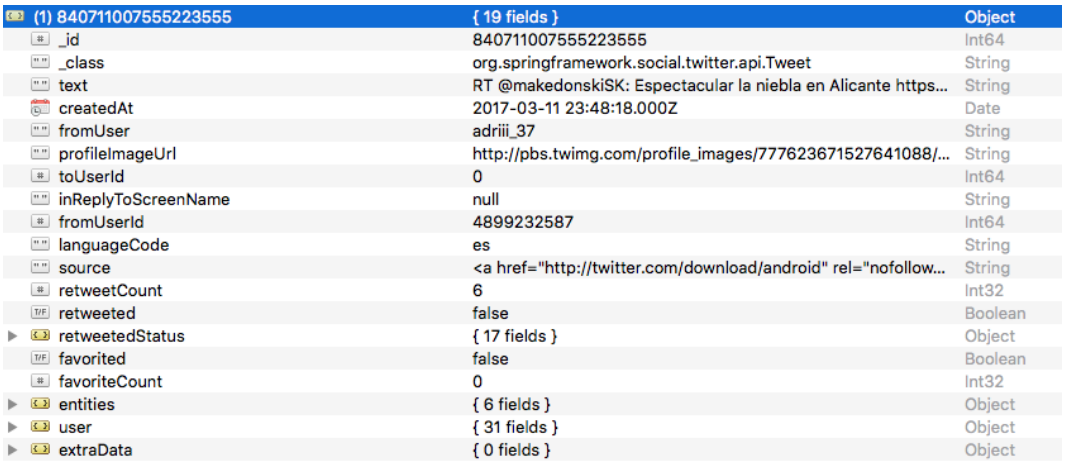
6.1. Capa de Persistencia

En la capa de persistencia hemos elegido dos bases de datos. Una base de datos NoSql para el almacenamiento de datos en bruto. Esta base de datos está en la primera fase, almacena todos los tweets ya sean de una zona en concreto o bien de un usuario según el modo de búsqueda.

La base de datos que hemos elegido es **MongoDB**. Mongo está basada en documentos y desarrollada en C++.

Una de las grandes ventajas que ofrece a TweetMe es el hecho de trabajar y almacenar datos en formato JSON. Los tweets que devuelve el API son en este formato y pueden ser volcados

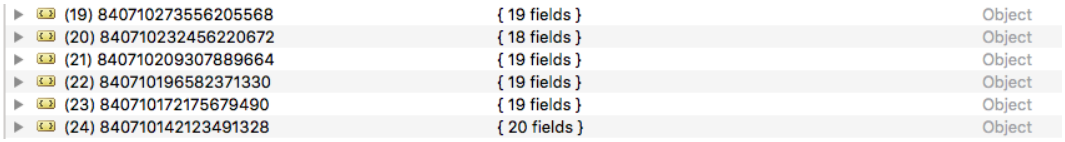
directamente en ella sin necesidad de utilizar mappers. JSON se basa en el principio de clave valor, es decir, cada atributo tiene asignado un valor. La Imagen 4 muestra la estructura básica de un documento almacenado en MongoDB.



(1) 840711007555223555	{ 19 fields }	Object
_id	840711007555223555	Int64
_class	org.springframework.social.twitter.api.Tweet	String
text	RT @makedonskiSK: Espectacular la niebla en Alicante https...	String
createdAt	2017-03-11 23:48:18.000Z	Date
fromUser	adriii_37	String
profileImageUrl	http://pbs.twimg.com/profile_images/777623671527641088/...	String
toUserId	0	Int64
inReplyToScreenName	null	String
fromUserId	4899232587	Int64
languageCode	es	String
source	<a href="http://twitter.com/download/android" rel="nofollow...	String
retweetCount	6	Int32
retweeted	false	Boolean
retweetedStatus	{ 17 fields }	Object
favorited	false	Boolean
favoriteCount	0	Int32
entities	{ 6 fields }	Object
user	{ 31 fields }	Object
extraData	{ 0 fields }	Object

Figura 5 – Detalle de un documento almacenado en MongoDB

El hecho de trabajar con JSON le ofrece a MongoDB la ventaja de tener una colección de documentos con distinta estructura, es decir, podemos insertar un tweet, documento, con menos campos que el resto de tweets, colección de documentos. Esto se puede observar en la Imagen 5 en el campo **fields.**



(19) 840710273556205568	{ 19 fields }	Object
(20) 840710232456220672	{ 18 fields }	Object
(21) 840710209307889664	{ 19 fields }	Object
(22) 840710196582371330	{ 19 fields }	Object
(23) 840710172175679490	{ 19 fields }	Object
(24) 840710142123491328	{ 20 fields }	Object

Figura 6 – Documentos con diferente número de campos

Las características que distinguen a MongoDB se podrían resumir en los siguientes puntos:

- **Alto rendimiento:** Mongo permite la opción de tener documentos anidados y así de esta forma se reduce el número de E/S, otra ventaja que aporta rendimiento es el hecho de trabajar con índices ya sea sobre arrays o subdocumentos.
- **Alta disponibilidad:** cuando hay un fallo en un nodo, Mongo automáticamente crea una réplica en otro nodo para poder seguir gestionando la información, esto es conocido como **réplica set.**
- **Escalado automático:** permite que se distribuya la información en otros clusters y así balancear la carga entre ellos.

Para finalizar este apartado cabe resaltar que hemos utilizado como cliente MongoDB, **Robomongo**[15]**.** Es un cliente con una interfaz sencilla y muy amigable con el usuario, permite una fácil visión de los datos que se están almacenando.

En cuanto a nuestra segunda bases de datos, es una base de datos relacional. Se entiende por base de datos relacional todas aquellas que cumplen el modelo relacional. En ellas se establecen relaciones entre entidades representadas como tablas, estas entidades almacenan datos que se relacionan con las de otras tablas.

[15] https://robomongo.org/

En la siguiente imagen se ilustra nuestro diagrama Entidad Relación.

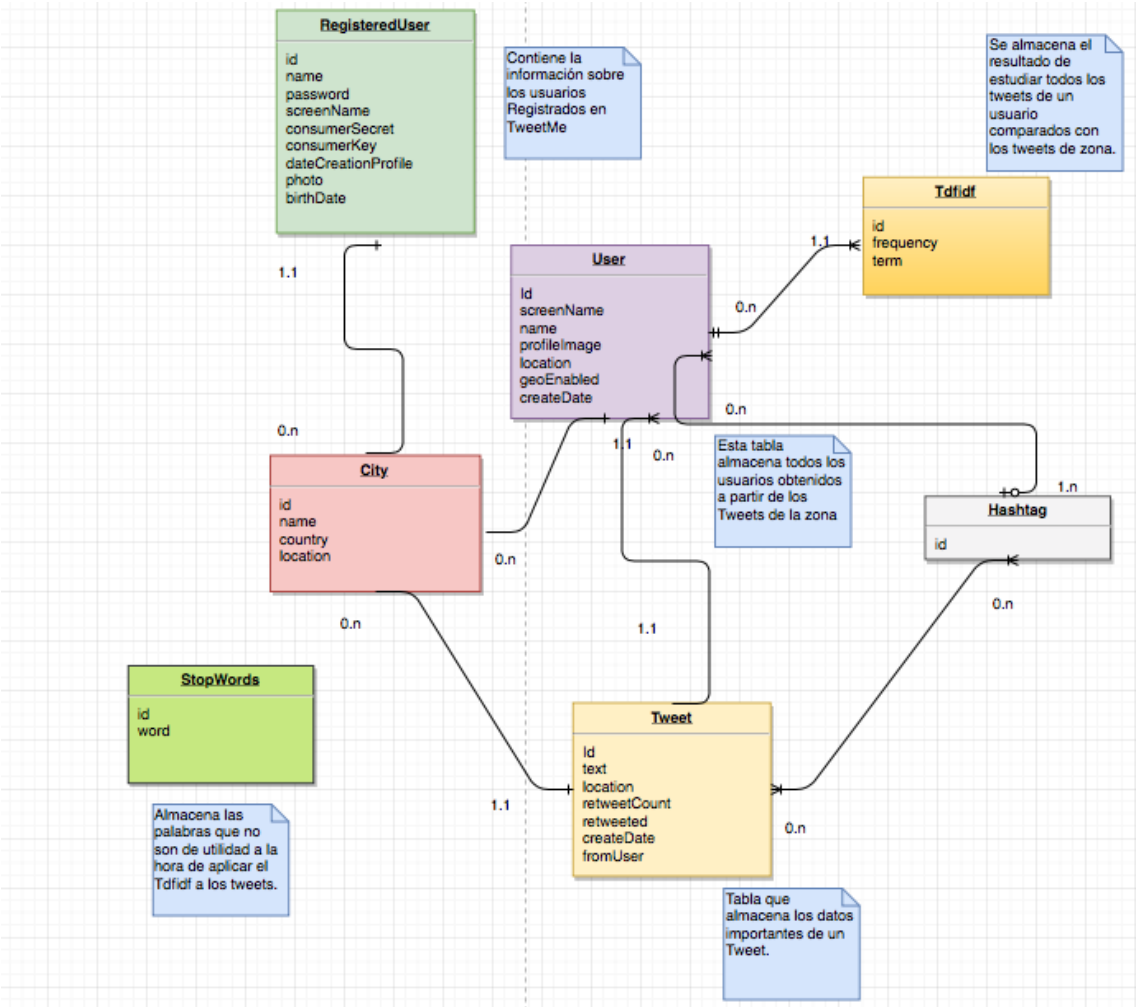


Figura 7 – Diagrama E/R de la bbdd relacional

Se puede afirmar que las entidades de mayor peso son aquellas que almacenan los tweets, usuarios de zona y el resultado de aplicar nuestro algoritmo TF-IDF.

Estas entidades o tablas almacenan datos resultado, es decir, almacenan justo aquello que deseamos mostrar en la vista, con esta información ya filtrada se obtiene un contenedor de datos preprocesados y de fácil accesibilidad permitiendo así que la tarea de recuperar información no requiere de múltiples consultas, consultas de larga duración y querys duplicadas.

Se ha elegido en todas ellas como clave primaria un índice. Los índices favorecen el rendimiento en las búsquedas y al ser incrementales cumplen la restricción de integridad referencial del modelo relacional.

Se han establecido relaciones entre entidades gracias a las claves ajenas, estas claves hacen referencia a cada clave primaria de la tabla a la que referencian.

El gestor de bases de datos por el que hemos optado es **PostgreSQL .** PostgreSQL es un sistema de gestión de bases de datos relacionales, es orientado a objetos ya que incluye características como la herencia, tipos de datos, reglas e integridad transaccional.

Algunas de sus ventajas son:

- Alta escalabilidad: se adapta al número de CPU's y la memoria disponible, gracias a esto es capaz de aguantar gran carga que puede duplicar la carga que puede soportar MySql.
- Rollbacks, transacciones y subconsultas: aporta PostgreSQL una mayor eficacia, esto es una carencia en el gestor de MySql.

Se ha elegido este gestor pensando en un futuro, pensando en crear una aplicación con gran cantidad de datos. Este hecho hace que MySql, aunque sea un gestor mucho más rápido, con bajo consumo de recursos a diferencia de PostgreSQL no puede abordar. En este caso nos hemos declinado por potencia frente velocidad.

PostgreSQL utiliza una arquitectura **cliente-servidor**, el servidor, **postgres,** es el administrador de archivos, el encargado de recibir las conexiones a las bases de datos y aplica sobre ellas las solicitudes procedentes de los clientes.

El cliente por su parte es el encargado de ejecutar las operaciones, no existe una única aplicación, en nuestro caso hemos utilizado **pgAdmin3**[16]. Al ser un gestor distribuido el cliente puede estar alojado en una máquina diferente al servidor, ambos se comunican a través del protocolo TCP/IP.

El servidor admite múltiples conexiones simultáneas, cada conexión se deriva en un hilo dejando el hilo principal atendiendo nuevas peticiones.

En TweetMe para mejorar el rendimiento de la aplicación web se ha añadido un pool de conexiones. Las aplicaciones web sufren muchas veces saturación y una caída de rendimiento si no hay una buena gestión de las conexiones que se realizan a la base de datos.

Para evitar que se realice una conexión a la base de datos de cada usuario hemos utilizado el llamado **Pool de conexiones**. El funcionamiento se basa en crear n conexiones en un primer momento a la base de datos. Estas están en modo escucha listas para ser usadas. Cuando la aplicación termina de usar una conexión, esta es devuelta al pool para su reutilización.

Pero la cuestión llega cuando se solicita un mayor número al establecido en un primer momento, en este caso se crean nuevas hasta el top que puede el pool y para finalizar en caso de que haya muchas conexiones esperando a nuevos clientes se van cerrando para reducir recursos.

En resumen, un pool de conexiones mejora el rendimiento de cualquier aplicación ya que minimiza el tiempo necesario para abrir una conexión a la base de datos.

6.2. Capa de Negocio

Para el desarrollo de esta parte tan fundamental se ha elegido como tecnología, el Framework Spring Boot. Spring Boot es una extensión del Framework Spring, un marco basado en Java para crear aplicaciones web.

A diferencia de muchos otros marcos, que se centran en una sola área, Spring Boot proporciona una amplia gama de características que abordan las necesidades de las empresas modernas a través de sus proyectos de ejemplo.

[16] https://www.pgadmin.org/

Spring Boot nace con la motivación de simplificar el proceso de configuración y despliegue de las aplicaciones creadas en Spring, ya que las aplicaciones desarrolladas en Spring requieren de muchos archivos XML de configuración.

Spring Boot se ha creado combinando bibliotecas de la plataforma de Spring y de terceros para que pueda empezar a trabajar con el mínimo esfuerzo.

La simplicidad de Spring Boot en cuanto a configuración es algo muy destacable a la hora de haberlo elegido como framework. Spring Boot se puede utilizar en un gran abanico de aplicaciones, ya sean tipo web o escritorio; tan solo hay que especificar o importar la dependencia en el fichero de configuración de Maven[17], pom.xml.

Hoy en día el marco de aplicaciones exige una gestión de todas las librerías y dependencias de nuestros proyectos, esto se debe a que una aplicación tiene dependencia de múltiples librerías y estas a su vez dependencia de otras, esto es un bucle de resolución de versiones que el desarrollador no puede controlar.

Para solucionar estos problemas nace **Maven.** Maven introduce el concepto de artefacto, un artefacto es una librería que además de incluir su propia información incluye la información necesaria sobre todas las librerías de las que depende. Esto se representa en clases.

Maven ha sido integrado en TweetMe como una herramienta fundamental de la gestión de todo el ciclo de vida del proyecto. Para definir todas las características de un artefacto se utiliza el archivo pom.xml, **Project Object Model**, en este fichero se detallan las dependencias y sus versiones, también todas las tareas para compilación, ejecución y tests del producto software.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocat

    <modelVersion>4.0.0</modelVersion>

    <groupId>com.teslem</groupId>
    <artifactId>tweetme</artifactId>
    <packaging>pom</packaging>
    <version>1.0-SNAPSHOT</version>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <spring-version>4.2.5.RELEASE</spring-version>
        <postgresql-version>9.1-901.jdbc4</postgresql-version>
        <spring-boot.version>1.3.5.RELEASE</spring-boot.version>
        <log4j-version>1.2.17</log4j-version>
        <commons-logging-version>1.2</commons-logging-version>
        <spring-data-commons-version>1.12.1.RELEASE</spring-data-commons-version>
        <commons-collection-version>3.2.1</commons-collection-version>
    </properties>
    <modules>
        <module>TweetMe-ModuleData</module>
        <module>TweetMe-ModulePorcessData</module>
        <module>TweetMe-ModuleAPI</module>
        <module>TweetMe-ModuleStarter</module>
    </modules>
    <build>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.3</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

Imagen 8 – POM padre de TweetMe

[17] https://maven.apache.org/

Como ya se mencionó con anterioridad TweetMe se distribuye en módulos, hemos creado un pom.xml para cada artefacto y un pom padre. En este pom padre se define el orden de complicación de sus hijos. Los módulos hijos se engloban en la etiqueta **</modules>** y están ordenados según la dependencia entre ellos.

Spring Boot y Maven son perfectos aliados para esta aplicación ya que todos los módulos que componen Spring Boot son fácilmente integrados en nuestro proyecto, solo hay que definir en el pom de cada módulo las dependencias que necesita. Algunos de los módulos utilizados en este proyecto son:

- **Spring MVC:** basado en Http y Servlets permitiendo extender y personalizar nuestras aplicaciones web y servicios Rest. Todas las peticiones se canalizan a través del front controller; Este canal es un Servlet propio de Spring, **DispatcherServlet**.

  El FrontController averigua a través de la URL qué controlador hay que llamar para gestionar la petición, **HandlerMapping.** Los resultados obtenidos se devuelven al Dispatcher en form de Modelo.

  Esta información en forma de modelo se muestra en la vista resuelta gracias a **ViewResolver.**

- **Contenedor de Inversión de Control, IoC:** este concepto se asocia a que cada objeto. Define sus propias dependencias a través de argumentos del constructor, es decir, en vez de que el propio objeto se encargue de instanciar las dependencias es el contenedor quien se encarga de ello a la hora de crear el bean. Es Spring quien hace el trabajo por nosotros.

  Las clases que abarcan el contenedor de Spring son:

  - **org.springframework.beans**: su interfaz BeanFactory ofrece el mecanismo de configuración para tratar todos los objetos. También posee una subinterfaz ApplicationContext, esta interfaz permite una fácil integración con algunas características del framework padre Spring, como es el manejo de eventos y contextos específicos para aplicaciones web.
  - **org.springframework.context**: agrega más funcionalidad muy específica del marco empresarial. las dos interfaces son representantes de contenedores de beans, solo que de tipos distintos.

Para poder comenzar con Spring Boot tan solo tenemos que combinarlo con **Gralde** o **Maven**[18]. En nuestro caso hemos optado por Maven debido a la experiencia previa.

6.2.1. Módulo API

Es el encargado de conectar directamente con el API de Twitter. Este módulo lanza las peticiones y obtiene toda la información que provee el API.

El API de Twitter es un conjunto de servicios REST que permiten acceder a todos los datos de Twiter. No solo podemos consultarlos, sino que también podemos modificarlos ya que el API permite realizar las mismas acciones que los usuarios de Twitter. Los datos que se envían y reciben en las peticiones viajan estructurados en formato JSON, permitiendo así una fácil integración en cualquier aplicación.

[18] https://maven.apache.org/

Aunque el acceso a los datos es público y abierto a todos los desarrolladores, es necesaria una autenticación para garantizar que no se hace un mal uso de dichos datos. Existen dos tipos de autenticación:

- Autenticación por aplicación:

  Este tipo de autenticación requiere que el usuario disponga de una cuenta en Twitter. Desde la interfaz de usuario de Twitter el usuario deberá registrar la aplicación (en nuestro caso TweetMe) y generar un par de tokens llamados **Consumer key** y **Consumer secret**. Estos dos tokens se envían en la cabecera de todas las peticiones REST.

  Twitter solo permite realizar operaciones de consulta de datos ya que los tokens pueden ser utilizados por diferentes usuarios impidiendo así registrar los cambios o inserciones que hizo cada usuario.

- OAuth:

  Este tipo de autenticación permite obtener tokens de autorización que son específicos de un usuario y que además tienen una duración limitada. Esto hace que se pueda identificar al usuario que está realizando la llamada al servicio. Como la vida de los tokens es corta, es necesario regenerarlos con bastante frecuencia. Solo el usuario que inicio el proceso de autenticación puede volver a generarlos, de esta forma se impide que usuarios no autorizados utilicen tokens con fines mal intencionados.

  Este tipo de autenticación da acceso a todos los servicios que ofrece el API REST.

Para evitar un uso excesivo de los servicios que pueda acabar saturando los servidores de Twitter, todos servicios dispones de limitaciones en cuanto a la cantidad de datos que se pueden utilizar en un determinado de periodo tiempo. Una vez excedido dicho límite, es necesario esperar cierto tiempo para poder seguir invocando a los servicios del API.

De forma muy inteligente, los servicios del API REST han sido agrupados según la funcionalidad a la que permiten acceder. Por tanto el API queda dividido en los siguientes grupos:

- Account
- Blocks
- Collections
- Direct Messages
- Favourites
- Friends
- Geo
- Help
- Lists
- Media
- Mutes
- Search
- Statuses
- Users

Hay que destacar los grupos Search, Users, Geo y Lists que son los que nos han proporcionado todos los datos necesarios para que TweetMe pueda funcionar.

Para simplificar el uso del API REST de Twitter se ha evitado utilizar el API directamente ya que integrar todas las llamadas REST junto con el proceso de autenticación explicado anteriormente resultaría una tarea complicada y que requeriría de una gran cantidad de tiempo. Por este motivo se ha decido utilizar el módulo **Spring Social**[19]que nos sirve como capa intermedia entre nuestra aplicación y el REST API de Twitter. Esta capa intermedia nos abstrae de las llamadas REST ya que estas quedan "ocultas" al programador, siendo el propio módulo el responsable de construir las peticiones y de convertir las respuestas a un formato fácilmente utilizable por la aplicación desarrollada.

Spring Social proporciona una factoría llamada **TwitterConnectionFactory** que es la encargada de realizar la autenticación con Twitter generando objectos que permiten interactuar con el API REST.

A continuación podemos observar en la Imagen 8 la distribución de este módulo en clases Java. Se ha definido una clase principal **TwitterConnector.** En ella se implementan los métodos principales que llaman al API.

Debido a que el API no permite obtener los usuarios de una zona concreta se han desarrollado dos operaciones.

Una primera operación **getTweetsByZone** que nos permite obtener los Tweets de una zona. Todos los Tweets recuperados son gestionados por el módulo Data. En cada Tweet viene detallado el usuario emisor del mismo.

Estos usuarios son el parámetro de la segunda operación **getUserTweets,** vuelve a recuperar tweets pero ya solo de un usuario específico. Cabe destacar que los tweets ya recuperados de la zona no se vuelven a insertar gracias al control de duplicados que tiene **MongoDB.**

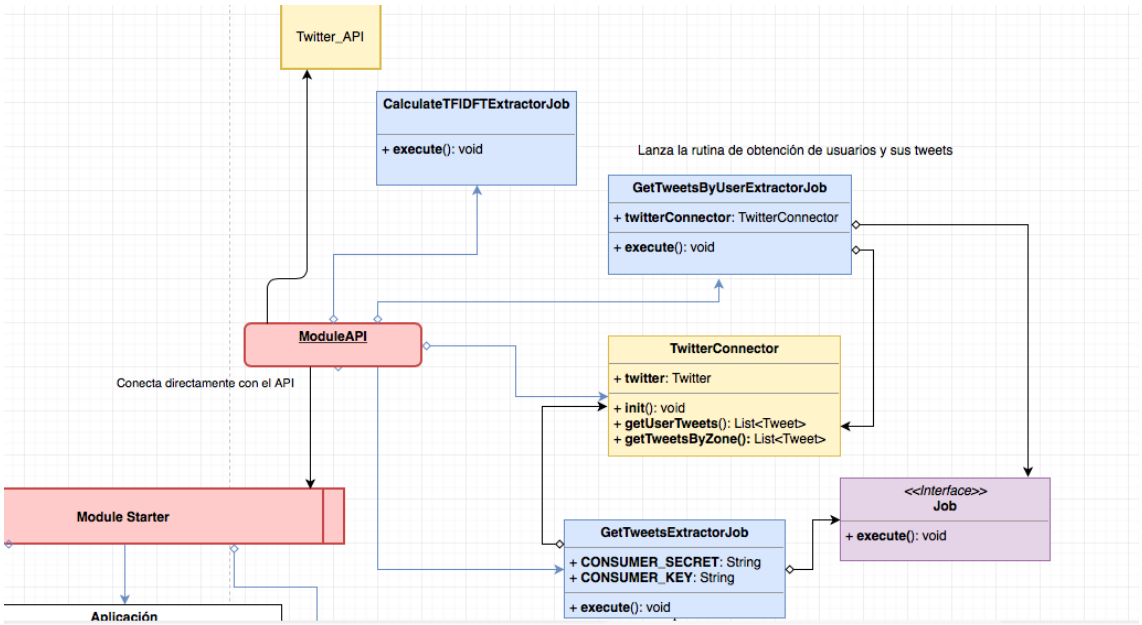


Imagen 9 – Diagrama de clases de la operación getTweetsByZone

[19] http://projects.spring.io/spring-social/

Debido a que el tiempo de vida de una petición HTTP no es lo suficientemente grande para esperar a que se recuperen los datos del API y sería incongruente mantener al usuario esperando la carga de la interfaz hasta entonces. Por este motivo se ha optado por utilizar **Quartz.**

Quartz [21]es un framework de scheduling open source que provee funcionalidad avanzada para la calendarización de tareas en Java. Sus principales características son:

- Agendar tareas en donde solo indiquemos la fecha y hora de ejecución y, a partir de ahí, definir una frecuencia de ejecución hasta agendar tareas mediante el poder de las expresiones de tipo Cron.
- Definir un medio de persistencia donde se almacene la información de tareas y sus agendas para poder darle la posibilidad al framework de recuperar dicha información ante fallas de la aplicación.
- Capacidad de trabajar en un ambiente de clusters.

Se han definido dentro del módulo tres clases que implementan la interfaz **Job,** en concreto el método **execute.** La clase **GetTweetsExtractorJob** recupera los tweets de zona llamando a nuestra clase TwitterConnector, la cual, a su vez, invoca al API. Una vez recuperados los tweets, estos son insertados en una base de datos MongoDB para su posterior procesamiento.

Aprovechando este Job se obtienen los usuarios almacenados en nuestra base de datos SQL. Estos usuarios, como ya hemos mencionado con anterioridad, son fruto de los tweets recuperados de la zona. Cada usuario es enviado a la clase TwitterConnector y de él se obtienen sus tweets que vuelven a almacenarse en la base de datos MongoDB.

En estos momentos ya se observa la interacción de los diferentes módulos que aún están por definir, como adelanto se puede resaltar que de todos los tweets de la MongoDB son trasladados a una base de datos SQL, pero en este caso solo quedándonos con los datos que nos interesan del tweet.

En cuanto a la clase **GetTweetsByUserExtractorJob,** cabe añadir que es una simplificación de la clase anterior. En este caso no nos interesa obtener todos los tweets de todos los usuarios, si no del usuario especificado en la búsqueda de quien interacciona con nuestra aplicación.

Finalmente, la última clase **CalculateTFIDFTExtractorJob** no interacciona con el API si no con nuestro módulo Process Data. Este Job se encarga de llamar al algoritmo que calcula el **TDIDF** para cada usuario. Toda la información obtenida es volcada a nuestra base de datos SQL para más tarde poder generar una nube de tags.

6.2.2. Modulo Data

Data es un módulo encargado de la inserción y extracción de datos, no posee lógica para tratarlos ni procesarlos, quedando este modo como un canal de datos o una pasarela entre las dos bases de datos y el resto de módulos.

Módulo Data posee todas las clases Java que actúan como Beans y Mappers y otros dos paquetes que albergan las consultas a las bases de datos, tanto a Mongo como a Postgresql.

[21] http://www.quartz-scheduler.org/

Las operaciones del paquete MongoDB se encargan del insertado de tweets obtenidos por el API en una base de datos no estructurada, NoSQL, y podrían verse como operaciones CRUD. Estas operaciones son invocadas por los Jobs anteriormente mencionados.

Cada Job invoca a métodos de estas clases para el almacenamiento de los datos en modo background.

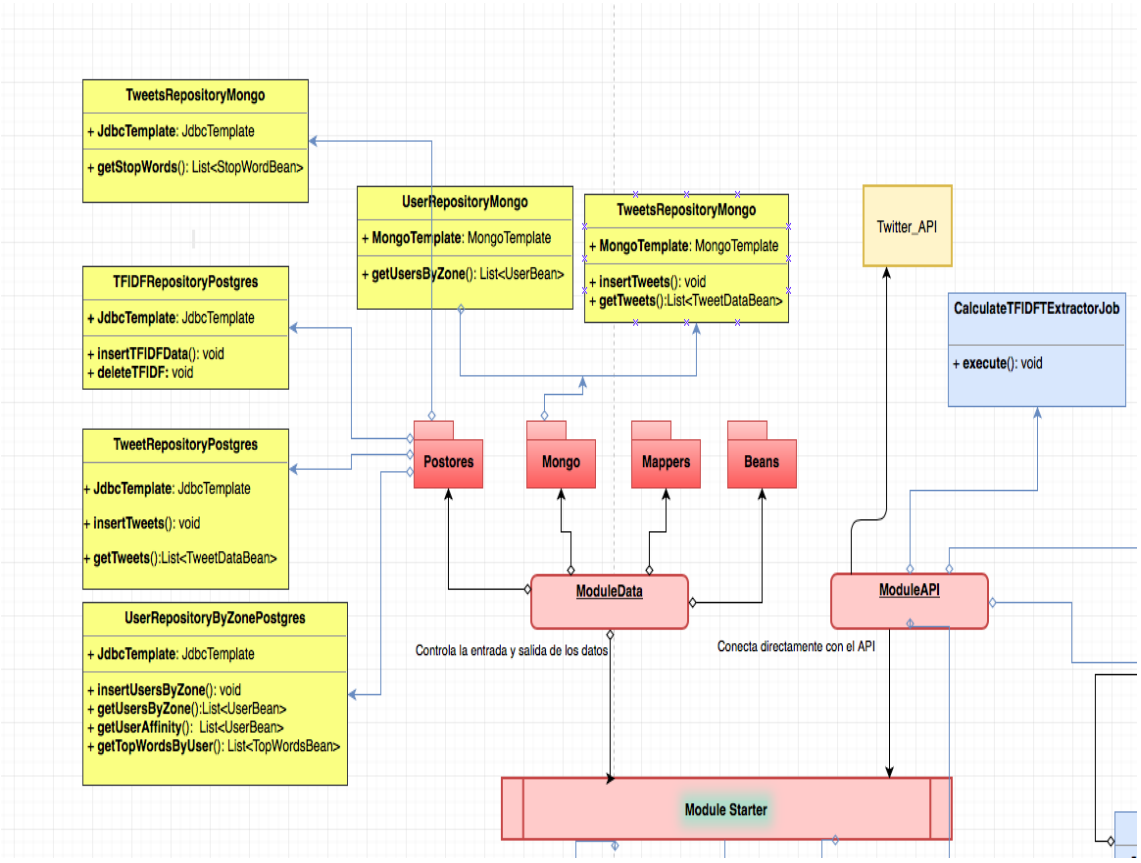


Figura 10 – Diagrama de clases del módulo Data

Ahora bien, para hacer más fácil la interacción de este módulo con una base de datos noSql, Spring Boot ofrece la dependencia **MongoTemplate.**

MongoTemplate[22] es un **ORM**, **Object-Relational Mapping**. Este concepto se refiere a transformar nuestras tablas en entidades, un ORM te permite mapear los datos de un objeto a un formato apto para la base de datos que estas utilizando.

Se crea una capa intermedia entre nuestra parte backend y la base de datos Mongo, esto permite abstraernos de la base datos y de alguna manera no preocuparnos por el motor de datos que se está utilizando en la capa de persistencia.

Con MongoTemplate ha sido muy fácil interactuar con la base de datos Mongo, ya que ofrece los métodos CRUD, (Create, Read, Update y Delete), básicos para tratarla a través de Java como lenguaje de alto nivel. Este módulo de Spring Boot no solo es fácil de usar sino que también ofrece seguridad a la capa de datos.

[22] http://projects.spring.io/spring-data-mongodb/

En cuanto a la interacción con la base de datos sql se ha elegido **JdbcTemplate,** ofrece algo similar que MongoTemplate pero enfocada hacia bases de datos sql. JdbcTemplate [23]nos abstrae de la base de datos que estamos utilizando con la condición de que sea una base de datos relacional.

JdbcTemplate simplifica la interconexión con la base de datos. Proporciona las herramientas básicas para realizar operaciones sobre la base de datos. Mediante el uso de Mappers, JdbcTemplate simplifica el proceso de convertir las tuplas devueltas por la base de datos relacional en objetos Java que son los que finalmente utilizamos en nuestra aplicación.

Una de las principales ventajas de utilizar JdbcTemplate frente a otros ORM es el hecho que podemos escribir las sentencias SQL directamente. De esta forma podemos sacar el máximo rendimiento a nuestra base de datos.

Todos estos detalles hacen del módulo data un componente central que ofrece operaciones que a simple vista no son muy tediosas pero aporta independencia entre los demás componentes de la capa de negocio y la capa de persistencia.

6.2.3. Módulo Process Data

Este módulo es el encargado, como su propio nombre indica, de procesar los datos. En los apartados anteriores se explicó como los datos son obtenidos de Twitter y almacenados en una base de datos. El módulo process data es el encargado de filtrar y aplicar todos los algoritmos necesarios a los datos para asociarlos a un contexto y convertirlos en información.

6.2.3.1. Filtrado y preprocesamiento

En la base de datos tenemos almacenados gran cantidad de datos referentes a los tweets publicados por los usuarios. Sin embargo, solo una pequeña parte de todos estos datos son de utilidad para su posterior procesamiento. Una de las principales funciones de este módulo consiste en descartar todos los datos innecesarios.

Otra de las funciones de este módulo consiste en ordenar y formatear, los datos útiles. Por este motivo podemos considerar que este módulo realiza un preprocesamiento de los datos lo cual simplifica y agiliza el posterior procesamiento.

Para realizar tanto el filtrado como el procesamiento se ha decidido hacer uso de los dos potentes gestores de base de datos que, como se explicó en apartados anteriores, forman parte de TweetMe. Dichos gestores de base de datos resultan idóneos ya que están específicamente diseñados para trabajar con datos. Se podría haber realizado estas dos tareas utilizando Java, lenguaje utilizado para desarrollar TweetMe, sin embargo se ha descartado esta opción porque el coste en tiempo de desarrollar la misma funcionalidad en Java es mucho mayor. Otro factor que ha inclinado la balanza del lado de los sistemas gestores de base de datos es el rendimiento, al estar específicamente diseñados para esta tarea su rendimiento es superior al de cualquier otro lenguaje de programación.

Como se comentó en apartados anteriores, es la base de datos noSQL la que contiene toda la información en bruto obtenida de Twitter. Filtrar los datos de forma que solo obtengamos los necesarios es una tarea bastante sencilla, basta con ejecutar una query en la que hacemos uso de las proyecciones. Una proyección se utiliza para devolver un conjunto concreto de campos de

[23] https://docs.spring.io/spring-framework/docs/current/spring-framework-reference/html/jdbc.html

un documento. De esta forma, sólo aquellos datos que nos son realmente útiles salen de la base de datos.

6.2.3.2. Cálculo de TF-IDF

En secciones anteriores se comentó que TweetMe aplica el algoritmo TF-IDF para obtener los términos más relevantes y que para ello se hace uso de la librería Lucene. Es en este módulo donde se realiza dicho cálculo utilizando la mencionada librería.

No ha sido una labor sencilla integrar Lucene por dos motivos:

- La librería está diseñada para leer documentos de archivos de texto. Sin embargo, en nuestra aplicación toda la información está almacenada en una base de datos. Además, se ha optado por definir como documento del corpus la suma de todos los tweets de un usuario. Por este motivo ha sido necesario realizar un algoritmo que extraiga y construya archivos de texto que se almacenan de forma temporal en disco para que Lucene los pueda procesar. Es en este punto donde se ha tenido que implementar un mecanismo que elimine de los tweets las denominadas palabras vacías. Las palabras vacías son aquellas palabras que no tienen significado como artículos, preposiciones, pronombres, etc.
- El resultado que genera Lucene se guarda a su vez en un fichero de texto. Es necesario, leer este archivo y volcar su contenido en la base de datos. Este paso requiere de un parseo del fichero para poder construir las sentencias SQL necesarias para insertar el resultado en la base de datos.

6.2.3.3. Análsis de sentimientos

Una de las funcionalidades de TweetMe es la de analizar si un tweet contiene texto que se puede interpretar como positivo o negativo. Para ello se ha utilizado un REST API desarrollado por el Grupo de Procesamiento Del Lenguaje Natural y Sistemas de Información (GPLSI) de la Universidad de Alicante.

Desde TweetMe se invoca a este servicio enviándole los tweets de cada usuario. El resultado es almacenado en la base de datos para su posterior presentación al usuario. Para evitar colapsar el servicio con un exceso de llamadas, solo se procesan los tweets de los usuarios que se han sido requeridos por el usuario para su análisis. Además, si los tweets de ese usuario ya fueron analizados, estos no vuelven a ser procesados.

A modo de conclusión, el módulo process data es el encargado de obtener los datos de la base de datos noSQL, procesarlos y almacenarlos en la base de datos relacional. Queda claro que toda la lógica de la aplicación se encuentra localizada en este módulo. Por tanto este módulo se encaja en la capa de aplicación en el modelo de tres capas definido en apartados anteriores.

6.2.4. Módulo Starter

Este módulo aunque resulte ser el más pequeño en cuanto a líneas de código se refiere, juega un papel fundamental en la aplicación ya que es el lugar donde se encuentra alojado el método main. El módulo Starter es el módulo central de toda la aplicación, el encargado de hacer que todo funcione.

En este módulo es donde se encuentran definidos todos los endpoints de los servicios REST que son invocados para la extracción de datos de Twitter, el procesamiento de dichos datos y su posterior visualización de los resultados.

Como se mencionó anteriormente, TweetMe se ha desarrollado usando Spring. Este framework permite la creación de servicios REST de una forma sencilla. Todo gira en torno a los controllers. Un controller es una clase Java en la cual hay métodos que se ejecutan cuando una determinada URL es invocada. Estos controllers, reciben y generan datos en formato JSON. La lógica de estos controllers es sencilla ya que la mayor parte del código se localiza en los módulos anteriormente descritos.

En este módulo también se localizan todos los archivos de configuración necesaria para inicializar la aplicación (conexiones a base de datos, loggers, etc).

Otro punto a destacar es el de seguridad. TweetMe dispone de un sistema de autenticación por token que impide a usuarios no registrados acceder a los datos. Este sistema de autenticación está basado en el standard JWT [24](JSON Web Token) que genera tokens que contiene información encriptada acerca del usuario y de la validez del token, el cual expira pasado un determinado periodo de tiempo. La ventaja de utilizar este sistema de autenticación es que no es necesario acceder a la base de datos en cada petición para determinar los privilegios de un usuario, ya que el propio token contiene dicha información.

Además de JWT, se ha utilizado un módulo de Spring llamado Spring Security que, entre otras muchas cosas, permite filtrar las peticiones a los servicios REST ofrecidos por la aplicación. Estos filtros leen de la cabecera de la petición el token generado por JWT y determinan si ese usuario tiene suficientes privilegios para ejecutar el servicio solicitado.

6.3. Capa de Presentación

La capa de presentación, también conocida como interfaz de usuario, es la encargada de la interacción con el usuario. Aunque no suele ser la capa que más esfuerzo requiere en ser implementada, si que es clave para el éxito de cualquier aplicación. Un mal diseño de esta capa puede conducir al fracaso de toda la aplicación. La interfaz de usuario ha de ser ante todo amigable, de forma que sea muy fácil entender su funcionamiento y presente los datos de tal manera que el usuario sea capaz de interpretarlos sin gran esfuerzo.

Elegir la tecnología adecuada para el desarrollo de esta capa es algo que no debe de tomarse a la ligera ya que esto determinara en muchos casos el conjunto de usuarios que serán capaces de utilizar la aplicación. En el caso de TweetMe se ha intentado llegar al mayor número posible de usuarios. Por este motivo se ha decidido implementar la capa de presentación en forma de aplicación web, lo cual permite acceder a la aplicación tanto desde un PC como desde cualquier dispositivo móvil.

6.3.1. HTML5 y CSS3

HTML5 es el último estándar publicado por el W3C[25] (World Wide Web Consortium). Consiste en un lenguaje de programación basado en etiquetas que permite definir interfaces de usuario. Este lenguaje es interpretado por los navegadores permitiendo definir interfaces multiplataforma. Su uso no requiere de un compilador y ha de ser escrito en documentos de texto plano. HTML5 está basado en XML heredando de él todas sus características. Algunas de las principales características son:

[24] https://jwt.io/
[25] http://www.w3c.es/

- Uso de etiquetas para definir la forma en que el navegador ha de construir la interfaz. Cada etiqueta consta de una etiqueta de comienzo y una etiqueta de fin. En medio de estas dos etiquetas se pueden incluir más etiquetas o texto. Una vez interpretadas las etiquetas, estas se transforman en los elementos que componen la interfaz.
- Uso de atributos para modificar la funcionalidad de los elementos que componen la interfaz de usuario.
- Permite transformar la interfaz en un DOM (Document Object Model) permitiendo tratar el documento como si fuese un árbol. Esta característica resulta de gran valor como veremos más adelante.

CSS (Cascading Style Sheets) es un lenguaje de programación que define la forma en la que se deben de presentar los elementos de un lenguaje de etiquetas como HTML. Se denomina estilo a los elementos que sirven para definir la distribución, colores y tipos de letra que se utilizaran a la hora de visualizar el contenido. CSS se compone de un conjunto de estilos que pueden ser agrupados en clases. Al igual que HTML, CSS es un lenguaje interpretado que se escribe en documentos de texto plano y que pueden ser utilizados en diferentes plataformas. Las principales características de CSS son:

- Separación del contenido y presentación.
- Define la forma en que los diferentes navegadores han de mostrar los elementos HTML.
- Permite la reutilización de los estilos a lo largo de toda la web.
- Sirve como herramienta para la unificación del diseño.
- Los estilos se pueden definir en un documento a parte haciendo así que en el código HTML quede mucho más claro.

Aunque no es un requisito indispensable, sí que es un muy aconsejable utilizar CSS3 junto con HTML5.

6.3.2. AngularJS 1.x

Es un framework de código abierto escrito en Javascript que se utiliza para el desarrollo de la parte front-end de aplicaciones web. Está implementado sobre un patrón Modelo-Vista-Controlador (MVC) y un patrón Modelo-Vista-Vista-Modelo (MVVM). Estos patrones permiten la separación del desarrollo de interfaces de usuario de la lógica de negocio.

AngularJS fue diseñado para trabajar en conjunto con HTML. Para ello AngularJS[26] hace uso de característica que aporta HTML que permite transformar la interfaz en un árbol DOM. Se considera que AngularJS es una tecnología de cliente, es decir, que se ejecuta directamente sobre el hardware del usuario que accede a la aplicación. Esta característica permite dividir el desarrollo de las aplicaciones en parte cliente y parte servidor permitiendo el desarrollo en paralelo de ambas partes.

Los tres pilares básicos sobre los que se sustenta AngularJS son:

- Vista: Se compone de HTML, CSS y atributos específicos de AngularJS que se incluyen en las etiquetas HTML.
- Controlador: Contiene la lógica de negocio.
- Scope: Representa el modelo de datos y además sirve de punto de unión entre la vista y el controlador. El scope implementa el denominado two-way data binging que detecta los cambios hechos en las variables del scope y actualiza el HTML en consecuencia.

[26] https://angularjs.org/

Si hay una característica que mejor puede definir lo que representa AngularJS es el concepto SPA o Simple Page Application.

Una SPA es una aplicación web que se implementa sobre una única página web. A diferencia de otros frameworks, AngularJS no navega de una página a otra para cambiar la estructura de la página. El mecanismo que utiliza AngularJS para no requerir de navegación consiste en modificar el árbol DOM que construye la página añadiendo, eliminando y modificando los diferentes nodos que componen el DOM. De esta forma se consiguen interfaces de usuario mucho más atractivas ya que al no recargar la página el usuario siempre tiene ante sus ojos una vista bien formada.

Por tanto, utilizar AngularJS mejora significativamente la experiencia de usuario.

7. Casos de estudio

En este apartado vamos a detallar un pequeño manual de funcionamiento de la aplicación.

Como hemos comentado en secciones anteriores, TweetMe dispone de dos interfaces principales después de hacer login en la aplicación.

7.1. Interfaz de Zona

Esta interfaz se corresponde con la primera a la que accedemos después de hacer login. Está dividida en secciones, en la primera de ella encontramos la portada principal, en la que aparece una nube de tags que corresponde a los resultados de aplicar nuestro algoritmo TF-IDF a toda la colección de tweets de zona, en esta nube se resaltan los términos de mayor valor de frecuencia obtenidos después de aplicar el algoritmo.

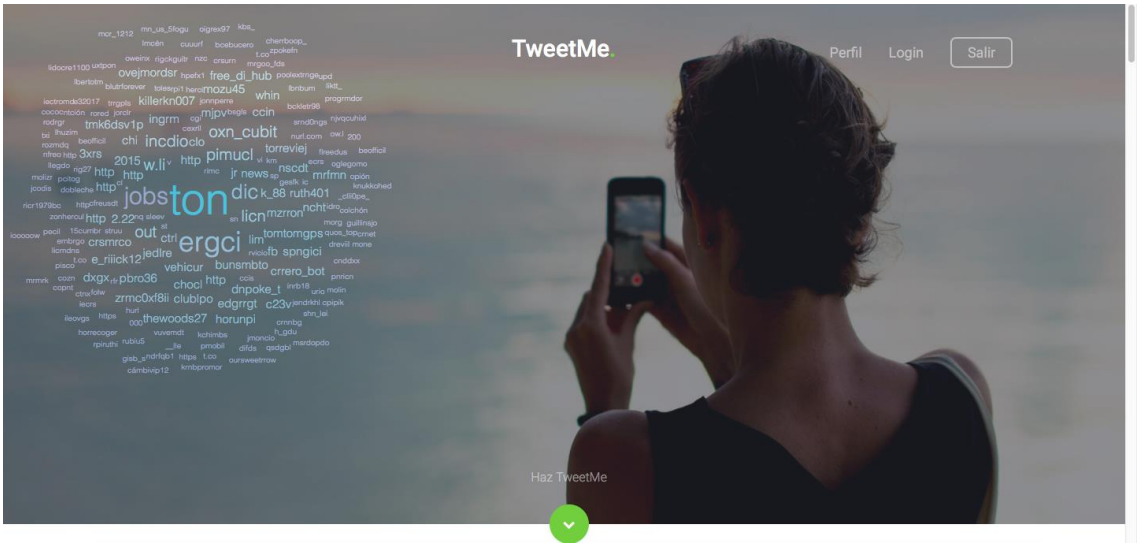


Figura 11 – Captura interfaz zona-1

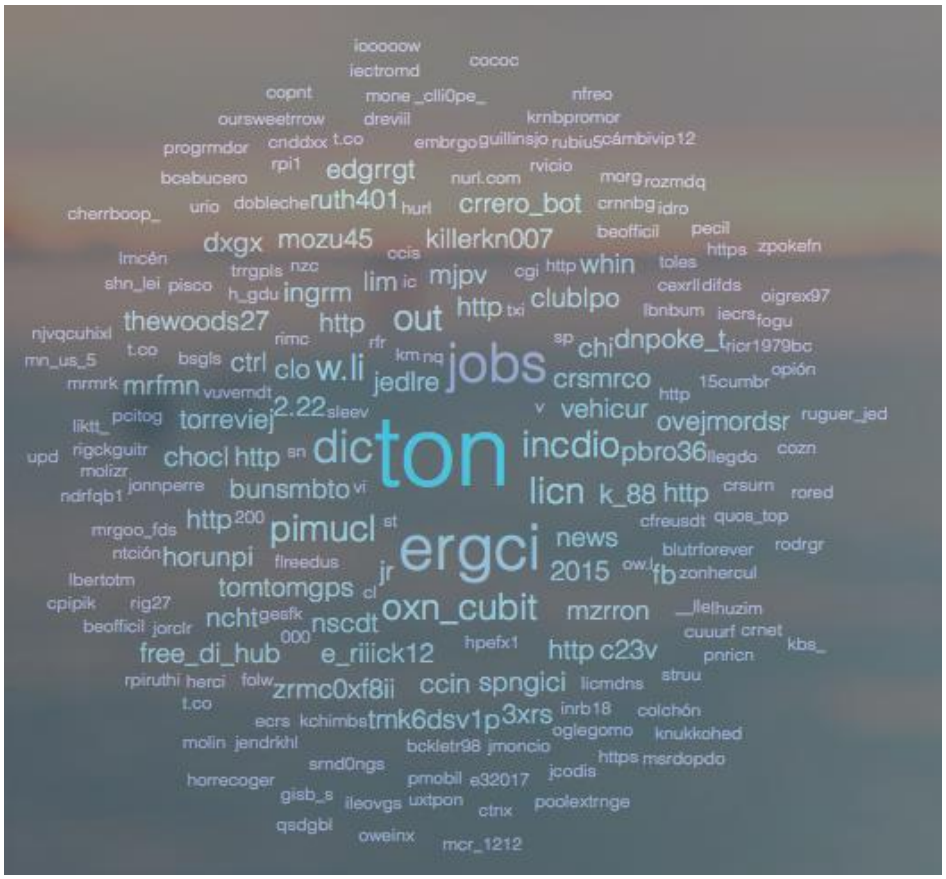


Figura 12 – Captura interfaz nube

El algoritmo se ha aplicado alrededor de **3 millones** de tweets y se han obtenido **3968** términos relevantes, en la nube de tag de la *Imagen 12* no se muestran todos los términos si no aquellos con una frecuencia significante. En la *Imagen 13* se ilustra un conjunto de estos términos y su frecuencia asociada.

term character varying	frequency double precision	user_id bigint	id bigint
ton	382.534912109375	11625111	2850427
ergci	249.420166015625	11625232	2874886
jobs	175.479354858398	11625783	3001044
dic	137.789901733398	11625232	2876074
pimucl	119.17236328125	11625727	2986429
w.li	112.07666015625	11625404	2919101
out	103.757118225098	11625783	3001363
incdio	103.586326599121	11625232	2875995
licn	103.088279724121	11625111	2850432
oxn_cubit	102.863067626953	11626015	3030889
http	99.0389022827148	11625477	2936168
bunsmbto	93.3467178344727	11625753	2992609
2.22	88.6841659545898	11625111	2850428
http	87.5025787353516	11625051	2834728
news	84.8060836791992	11625274	2889435
vehicur	84.7696762084961	11625232	2874840
mjpv	83.4240417480469	11625389	2913668
2015	82.692268371582	11625389	2913463
tomtomgps	81.0154190063477	11626275	3071792
jedlre	81.0154190063477	11625706	2982769
clo	80.8011779785156	11625232	2874948
http	79.2497711181641	11625671	2975852
crsmrco	79.1939849853516	11626324	3084262
clublpo	77.8091278076172	11625919	3019690
lim	77.6672439575195	11625232	2876071

Figura 13 – Captura BBDD sql.

A continuación de esta primera parte se encuentra el buscador. El buscador nos permite buscar cualquier usuario de zona por su "nickname" para poder posteriormente analizar aquello que publica y aplicar el algoritmo anterior a sus tweets conjuntamente con los tweets de zona.

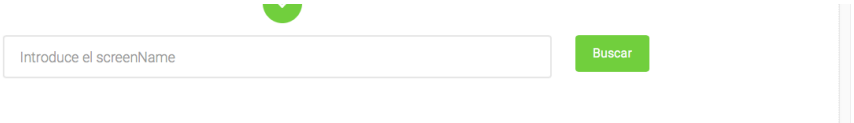


Figura 14 – Captura buscador TweetMe.

Otra funcionalidad que se lista en esta interfaz son aquellos usuarios, con mayor número de tweets de la zona, más términos posee en la tabla obtenida después de aplicar el algoritmo a todos los tweets de zona y finalmente el usuario que más seguidores posee.

Figura 15 – Captura Interfaz de zona.

Y finalmente, una sección donde se listan todos los usuarios de zona, es una sección dinámica ya que se van listando de manera aleatoria. Podemos hacer doble click sobre cada uno de ellos y es como si los hubieramos buscado como hemos comentado en la *Imagen 14.*

Se han obtenido hasta ahora alrededor de **2000** usuarios en la ciudad de Alicante.

Figura 16 – Captura Interfaz de usuarios de zona.

7.2. Interfaz de Usuario Buscado

Esta interfaz constituye la parte principal de la aplicación y en ella se detalla todo el análisis sobre un usuario buscado de una zona concreta.
Para nuestro caso de estudio hemos elegido al primer usuario obtenido por la herramienta, con nickname "**Adriii_37.**

Figura 17 – Captura Interfaz Usuario adriii_37.

En esta interfaz hacemos un análisis de todos los tweets obtenidos de este usuario y con la herramienta proporcionada por la Universidad de Alicante podemos saber si un tweet es positivo o negativo.

De adriii_37 hemos obtenido tweets desde **25/08/2016** hasta **11/06/2017**, de todos ellos hay **194** de carácter positivo y **218** de carácter negativo.

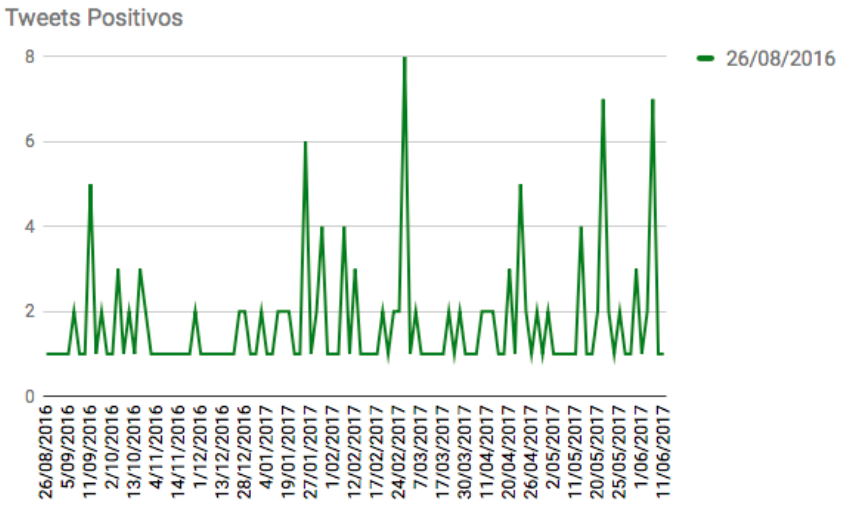


Figura 18 – Captura grafica execel Positivos.

Como se puede observar en el gráfico los día que más tweets positivos ha publicado ha sido el día **26/02/2017,** 8 tweets en concreto ese día.

En TweetMe hemos utilizado gráficos para ilustrar estos resultados por año y hacer más amigable la interfaz con el usuario. La herramienta ilustra los tweets por días y por años, en la *Imagen 19* como se puede observar están listados todos los del año 2016 y si desplazamos el ratón sobre la gráfica podemos ver la cantidad de tweets por día. La gráfica de color rosado corresponde a tweets negativos y la de color azulado a tweets positivos.

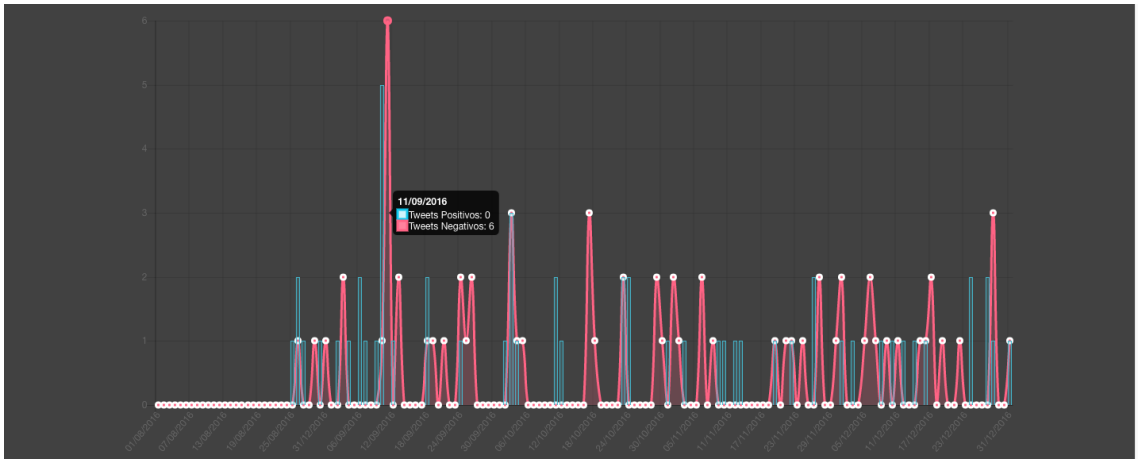


Figura 19 – Captura TweetMe tweets negativos y positivos.

En cuanto a los tweets negativos obtenidos para este usuario han sido **215** y se puede observar que el día que más tweets ha publicado ha sido el **01/03/2017**, un total de 9 tweets.

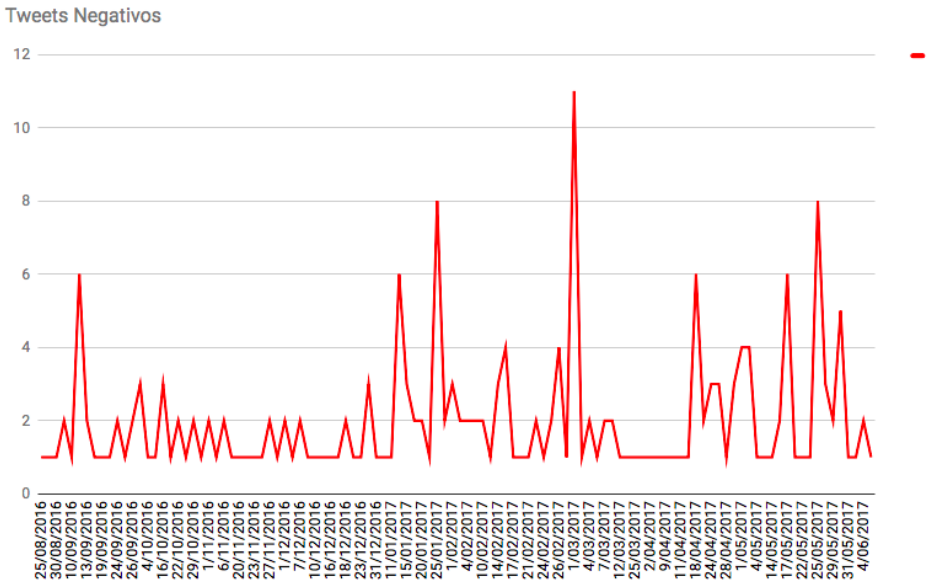


Figura 20 – Captura excel tweets negativos.

Para que el usuario pueda visualizar todos los tweets se ha creado un timeline donde se listan todos ellos. En esta interfaz están destacados aquellos positivos y aquellos negativos.

Figura 21 – Captura timeline TweetMe.

En resumen, adriii_37 ha publicado un mayor número de tweets negativos que positivos en todo este periodo de tiempo. Cabe destacar que esto no es algo exacto ya que el análisis del lenguaje natural es una tarea muy compleja.

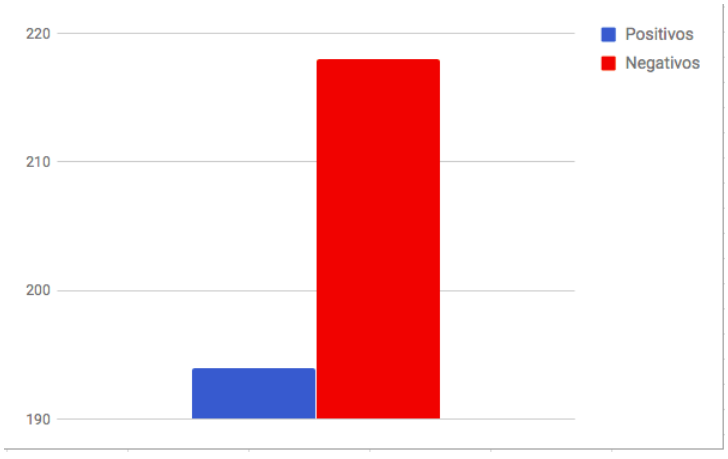


Figura 22 – Captura comparacion tweets negtivos y positivos.

TweetMe aprovecha la geolocalización del último tweet publicado por un usuario para dar una aproximación de dónde ha estado, por este motivo hay un mapa donde se señala la localización de dicho usuario.

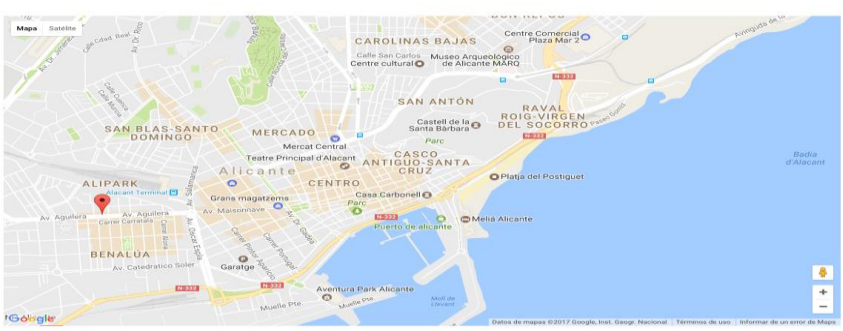


Figura 23 – Captura mapa localización del usuario.

Aprovechando toda la información que viene en los tweets hemos podido añadir una mayor funcionalidad a TweetMe. Detalles como los que se observan en la *Imagen 24* nos permite concluir que adriii_37 es un usuario que genera menor número de tweets propios y la mayoría de sus tweets son retweets.

El último tweet recogido nos brinda información no solo de la ubicación de un usuario sino también de la hora en la que ha estado en ese lugar. Los tweets nos permiten tener geolocalizados a los usuarios de una zona.

Figura 24 – Captura mapa localización del usuario.

Y para finalizar esta sección hemos incluido una representación gráfica de aquellas palabras de mayor relevancia obtenidas tras aplicar el algoritmo TF-IDF a todos los tweets de un usuario buscado, en ese caso adriii_37, y los tweets de su zona. Esta gráfica representa lo mismo que la nube de tags.

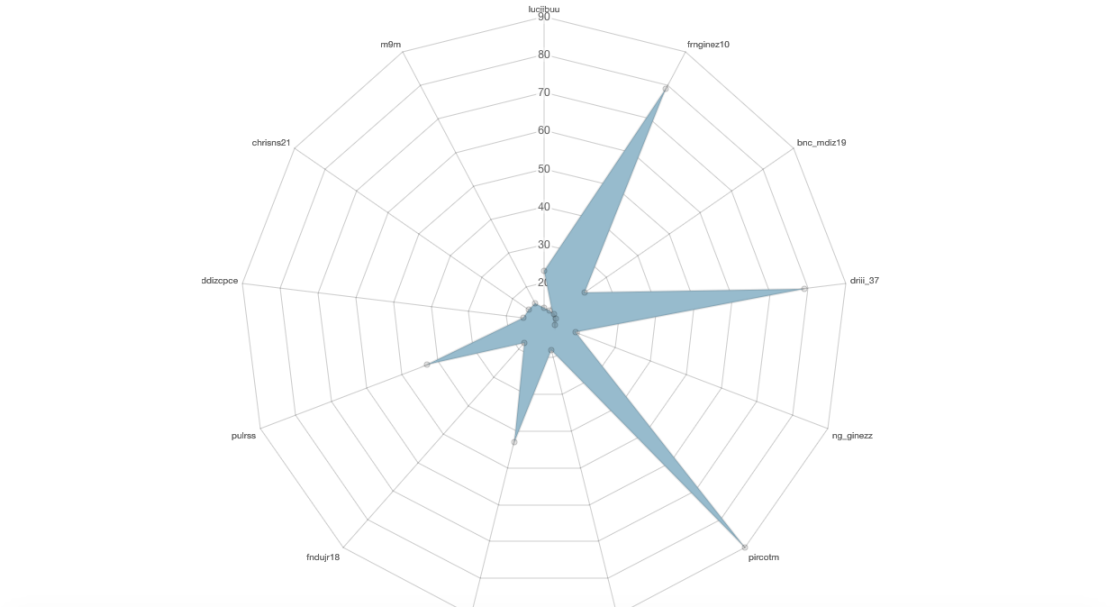


Figura 25 – Captura gráfica adri TF-IDF.

Adri tiene **258** términos de relevancia, algunos de ellos se muestran en la siguiente figura.

term character varying	frequency double precision	user_id bigint	id bigint
luciibuu	28.1450958251953	11625332	2897043
frnginez10	23.7869472503662	11625332	2896841
bnc_mdiz19	23.7869472503662	11625332	2896968
driii_37	19.9015865325928	11625332	2897021
ng_ginezz	18.4252891540527	11625332	2896964
pircot	18.4252891540527	11625332	2896790
fri_f	15.0441856384277	11625332	2897006
junnsnchez25	15.0441856384277	11625332	2896810
fndujr18	15.0441856384277	11625332	2896939
pulrss	13.2116041183472	11625332	2896827
ddizcpce	13.0286464691162	11625332	2896812
chrisns21	13.0286464691162	11625332	2896895
m9m	13.0286464691162	11625332	2896884
pu_lres	13.0286464691162	11625332	2896889
cudidm25	12.3263607025146	11625332	2897039
e_nrro11	11.4415845870972	11625332	2897023
luiito_7	10.6378450393677	11625332	2896928
igrblm	10.6378450393677	11625332	2896990
jorgee_sg8	10.6378450393677	11625332	2897016
molintoo	10.6378450393677	11625332	2897028

Figura 26 – Captura resultados Adri cálculo TF-IDF.

8. Conclusiones

TweetMe se ha desarrollado como una aplicación web para recoger toda la información que publicamos en Twitter. Es una herramienta capaz de analizar tweets, y extraer información útil para poder analizar rasgos personales de los usuarios. Todo ello gracias a la aplicación de Algoritmos de Procesamiento del Lenguaje Natural.

Para acercar el desarrollo de TweetMe a las metodologías ágiles se ha intentado utilizar herramienta de planificación como Trello, de generación de diagramas como Draw.io y gestión de versiones como Git. Con todas estas herramientas se puede visualizar todo el proceso de construcción de la herramienta día a día.

En este proyecto se ha apostado por tecnologías y frameworks de desarrollo de código abierto como Spring Boot en la capa de negocio, MongoDB y PostgreSQL en la capa de persistencia y AngularJS para la capa de presentación. Todas estas tecnologías aportan las herramientas suficientes que han hecho posible el desarrollo de TweetMe.

La intención final de esta aplicación es la de crear una comunidad de desarrolladores que amplíen la funcionalidad ya que, como se comentó en apartados anteriores, solo estamos usando una pequeña cantidad de todos los datos que nos ofrece Twitter.

La arquitectura del proyecto es compleja, de carácter empresarial. TweetMe está pensado para ser ampliado tanto en funcionalidad como extendido a aplicaciones móviles. Se pretende crear una aplicación móvil aprovechando la misma capa de negocio y creando únicamente una interfaz nativa de tecnologías móviles.

Crear una aplicación móvil aporta mucho valor a este proyecto ya que toda nuestra sociedad tiende hoy en día a acceder a todo el contenido digital a través de un smartphone. Por ello una de las primeras opciones de un futuro próximo es hacer de TweetMe una aplicación móvil dejando atrás la web.

Desde un punto de vista personal este proyecto ha sido un reto ya que muchas de las tecnologías por las que he apostado no tenía conocimiento previo de ellas, esto me ha permitido mejorar mucho como desarrollador. He apostado por tecnologías que se utilizan actualmente en el ámbito empresarial para obtener una aplicación distribuida. Cada fase de desarrollo me he enfrentado a múltiples problemas que han ayudado a mi formación en las diferentes tecnologías que antes desconocía.

TweetMe para mí siempre será un punto y aparte en mi etapa educativa, he dedicado mucho tiempo y esfuerzo pero en ningún momento me he desanimado para no continuar. Desde el día de la propuesta me ha parecido un proyecto de gran interés donde podría mejorar mucho mi perfil como profesional y aprender un gran abanico de tecnologías.

9. Agradecimientos

Me gustaría dedicar este apartado a mi director de proyecto, David Tomas Diaz, por haberme ofrecido la oportunidad de colaborar contigo en este proyecto, gracias por el soporte continuo tanto en ideas, documentación y sobre todo en ánimos para desarrollar este proyecto.

Te agradezco la confianza que me has brindado en todas las fases de desarrollo de este proyecto y por ser tan comprensivo con los plazos y entregas.

10. Bibliografía

https://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/#getting-started-introducing-spring-boot
http://docs.spring.io/spring-social-twitter/docs/1.0.5.RELEASE/reference/html/overview.html#overview-introduction
https://sg.com.mx/revista/18/una-introduccion-quartz-calendarizacion-tareas-java#.WRceKFKxj-Y
https://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-nosql.html
http://www.arquitecturajava.com/servicios-rest/
http://instintobinario.com/arquitectura-en-tres-capas/
http://oa.upm.es/38731/1/TFG_Federico_Gutierrez_Faraoni.pdf
http://www.jtech.ua.es/j2ee/publico/spring-2012-13/sesion03-apuntes.html
http://www.javatutoriales.com/2010/12/contenedores-de-ioc-e-inyeccion-de.html
https://programarfacil.com/blog/que-es-un-orm/
https://es.wikipedia.org/wiki/Maven
https://www.genbetadev.com/java-j2ee/que-es-maven
http://www.jtech.ua.es/j2ee/publico/spring-2012-13/sesion02-apuntes.html
http://www.manualweb.net/mongodb/que-es-mongodb/
https://anotherdayanotherbug.wordpress.com/2015/01/07/spring-boot-series-autoconfiguracion-de-jackson/
http://www.manualweb.net/mongodb/que-es-mongodb/
http://www.uprh.edu/adem/Base%20de%20datos%20relacional.pdf
https://danielpecos.com/documents/postgresql-vs-mysql/
http://www.barriblog.com/2010/07/lo-que-siempre-quiso-saber-del-api-de-twitter-y-nunca-se-atrevio-a-preguntar/
http://blog.elogia.net/historia-redes-sociales-origen/
http://www.yoseomarketing.com/blog/que-son-las-redes-sociales-para-que-se-utilizan/