Trabajo Fin de Grado en Ingeniería Informática

Análisis de actividad en redes sociales

Rubén Gabás Celimendiz

Director:
Fernando Bobillo Ortega

Departamento de Informática e Ingeniería de Sistemas
Escuela de Ingeniería y Arquitectura
Universidad de Zaragoza

Noviembre de 2019
Curso 2018/2019

Agradecimientos

Quiero dedicar este Trabajo Fin de Grado
a mis padres por su apoyo incondicional,
y a Fernando por la oportunidad que me ha brindado.

Resumen

Este documento describe la creación de un sistema de análisis de métricas, comparación e histórico sobre redes sociales (como Twitter, Instagram, Facebook ...). Se ha desarrollado este sistema para Twitter con un modelo de datos flexible, para la fácil extensión de la aplicación. La idea a largo plazo sería incluir otras redes sociales, como por ejemplo las mencionadas anteriormente.

La popularización de las redes sociales, que cuentan con millones de usuarios e interacciones al día, ha hecho que empresas, marcas y personajes famosos hagan de estas un marco de negocio, ya sea por facilidad de difusión de contenido, cercanía a los clientes, atención al cliente, campañas de marketing... Hay cuentas de usuarios, marcas y empresas que generan mucha información diariamente, como publicaciones e interacción con otros usuarios (lo que supone un gran marco de negocio) y más datos medibles como el tipo de publicaciones (texto, archivos multimedia, interacciones...).

Existe un gran volumen de información más allá de la percibida por los usuarios. Esta problemática motiva la creación de un analizador capaz de realizar no solo una simple labor de extracción de datos, sino también de calcular para todas las publicaciones los valores de algunas métricas relevantes.

Además de recuperar estos datos en el sistema, se guardarán las analíticas y la información necesaria para poder consultarse en cualquier momento, ahorrando tiempo de ejecución y permitiendo su uso a posteriori. Dicha capacidad, permite estudiar la evolución en el tiempo del estado de los usuarios y sus publicaciones, o incluso comparar distintas cuentas de usuario y los datos de las mismas.

Se han analizado varios casos de uso para demostrar la utilidad de esta solución y cómo aplicarlos en ciertos negocios para ayudar a evaluar el uso de las redes de su competencia.

Abstract

This document describes the creation of a metric analysis comparison and historical system on social networks (Ex: Twitter, Instagram, Facebook...). The system has been developed for Twitter with a data model for easy application extension. The long-term idea would be to include other social networks, such as those mentioned above.

The popularization of social networks, which have millions of users and interactions a day, a fact that companies, brands and renowned people make these a business framework, whether for dissemination of content ease, proximity to customers, customer support, marketing campaigns ... There are user accounts, brands and companies that generate lots of information for daily use, such as publications and impact on social networks, users in contact with the accounts mentioned above (which is a great business framework), and more measurable data such as the type of publications (text, multimedia files, interactions ...).

There are large volumes of information beyond that perceived by users, which does not pose a simple extraction, since publications should be analyzed by marking the most relevant data on them. This problem arises the idea of creating an analyzer capable of performing data extraction work, in an acceptable time, and visually present it in an orderly and understandable way for the user.

Besides storing this data in the system, the analytics and the necessary information will be able to be recovered at any time, saving execution time and being able to be used later to study the evolution in time of the state of users and their publications, or even compare different user accounts and their data.

Use cases have been analyzed to demonstrate the usefulness of this solution and how to apply it in certain businesses to help evaluate the use of your competitor networks.

Índice

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción y Objetivos

Este Trabajo Fin de Grado (TFG) responde a la necesidad existente de crear un sistema de almacenamiento, análisis y reporte de datos de ciertas redes sociales.

1.1. Contexto

El término de red social se define como el conjunto de relaciones entre miembros de un sistema social a diferentes dimensiones [Roc]. Dicho concepto cuenta con, aproximadamente, una antigüedad de unos 100 años [Wikh], En la actualidad, además del componente de comunicación entre individuos, las redes sociales se usan a su vez para analizar interacciones entre individuos, grupos, organizaciones, e incluso hasta sociedades enteras.

Fue en los años 90, con la popularización de la *World Wide Web* [Wikj],cuando el concepto de red social comienza a dar los primeros pasos hacia el mundo digital. La web denominada `classmates.com`, fue una de las primeras redes sociales concebidas digitalmente [Wika], cuya funcionalidad es buscar amigos desde la guardería hasta la universidad. Acercándose al concepto de red social que se tiene actualmente, *SixDegrees* (1997) fue una de las primeras redes basadas en el modelo del circulo social [Wikg].

Desde el inicio del milenio, siguiendo el concepto de circulo social, empiezan a surgir las redes sociales orientadas a la interacción entre los miembros: Friendster (2002), LinkedIn (2002), MySpace (2003) y Facebook (2004). Algunas de estas no sólo siguen teniendo éxito hoy en día, sino que se han convertido en uno de los sitios web en generar más trafico actualmente [Har]. Entre las más relevantes y las estudiadas para este TFG se encuentran Facebook[1], Twitter[2], Instagram[3] y LinkedIn[4].

[1] `https://www.facebook.com/`
[2] `https://twitter.com`
[3] `https://www.instagram.com`
[4] `https://linkedin.com`

Esta elección se justifica no sólo en la popularidad de cada plataforma [1.1], sino en el propósito y tipo de usuarios que las frecuentan. Por ejemplo, LinkedIn es un concepto de red social distinto al de las otras tres, al ser una red de ámbito profesional en la que se encuentran datos de ofertas de trabajo, datos de empresas, artículos públicos y miles de publicaciones por semana.

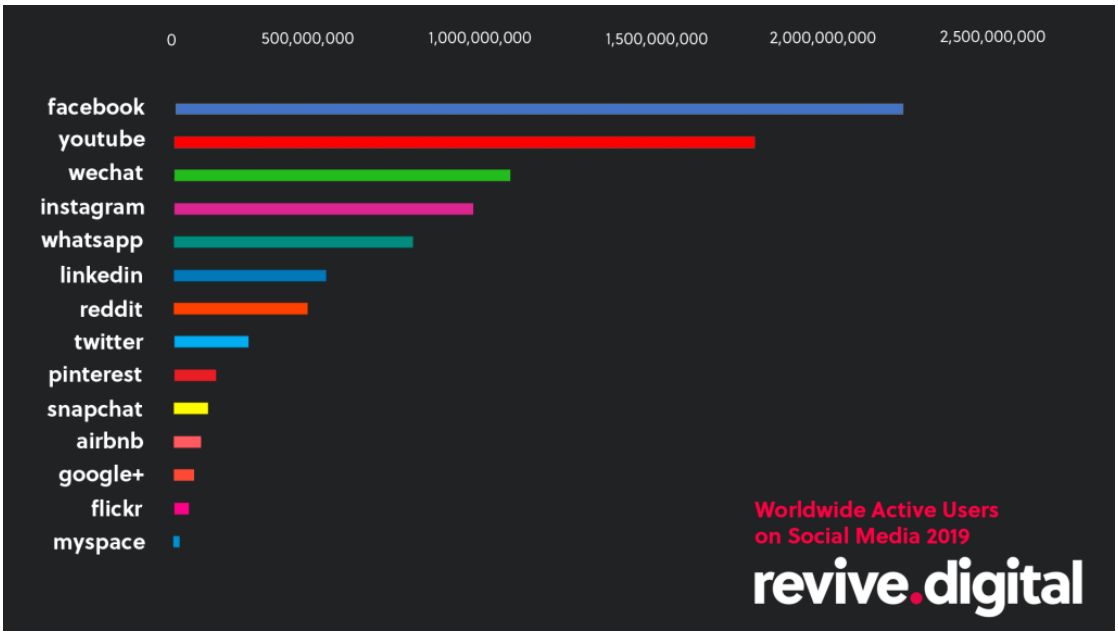

Figura 1.1: Volumen redes sociales

Fuente: [Rev]

Hoy en día muchas corporaciones tienen una marcada presencia en las redes sociales, siendo de gran importancia poder analizar el uso de las mismas y compararlo en distintos puntos del tiempo. Además, el hecho de que la actividad de un usuario dentro de una red social sea difícil o imposible de medir con las herramientas nativas de cada plataforma, ha originado un ecosistema de herramientas y empresas que intentan explotar dicha actividad [Caw]. Uno de los ejemplos más polémicos y recientes es la influencia que produjo la empresa Cambrigde Analytica sobre la campañas políticas de Estados Unidos [Cha].

Las plataformas sociales populares se han convertido en gigantes del marketing y ofrecen información valiosa a los negocios sobre sus clientes en gran medida de una forma gratuita para llegar a ellos [EN]. A continuación se resumen sus principales puntos clave:

- **Reunir información valiosa sobre los clientes**. Se pueden averiguar tendencias en los segmentos de población de los usuarios, quiénes son, cómo son y qué opinan de tu marca.

- **Aumentar el reconocimiento de la marca y la fidelidad de los clientes**. Si se tiene presencia en las redes sociales, será más fácil que los clientes puedan interactuar contigo, con lo que hay posibilidad de aumentar la retención y fidelidad de la marca.
- **Crear anuncios segmentados con resultados en tiempo real**. Los anuncios sociales son económicos y útiles para promocionar el negocio y distribuirlo, y ofrecer distintas opciones de campañas publicitarias para distintas audiencias. También se puede medir el resultado de estos anuncios en tiempo real.
- **Generar clientes y ventas**. Las redes sociales aumentan las ventas y la retención de clientes gracias a un buen servicio de atención al cliente e interacción frecuente.Según un estudio de investigación se considera a las redes sociales como una de las mejores formas de generar nuevas oportunidades de negocio [MHI].
- **Ofrecer valiosas experiencias para los clientes**. Los clientes de las empresas esperan que esas empresas estén en las redes sociales, las estadísticas muestran un 67 % de los clientes usan el servicio al cliente de las empresas en las redes sociales esperando atención 24/7.
- **Aumentar el tráfico de tu página y la posición en los buscadores**. Uno de los mayores beneficios de las redes sociales en los negocios es utilizarlas para incrementar el tráfico de tu página web. Las redes sociales no solamente te ayudan a dirigir a la gente hacia tu página web, sino que cuanto más compartan tus contenidos en las redes sociales, más alta será tu posición en buscadores.
- **Compartir contenido más rápido que nunca**. Antiguamente las empresas tenían dificultad para llegar a los clientes en el menor tiempo posible, con las redes sociales solo tienes que compartir el contenido que se necesite en las redes sociales de tu marca

Se ha mencionado la importancia que tienen las redes sociales en la experiencia del usuario y en el ámbito empresarial. Esta motivación, sumada a las limitaciones de funcionalidad y disponibilidad que se detallarán en el Capítulo 2, se ha decidido por desarrollar una plataforma de análisis con métricas a medida.

1.2. Objetivos

El objetivo principal de este Trabajo de Fin de Grado (TFG), es el desarrollo de un sistema de análisis, generación de informes, historial y comparación, sobre datos de usuarios de redes sociales. Se ha limitado el alcance del proyecto a Twitter por una serie de características que se comentarán más adelante, en el Capítulo 2.

- **Objetivo 1**: Analizar las distintas redes sociales más importantes (Twitter, Facebook, Instagram y Linkedin) e investigar sus distintas APIs[5].
- **Objetivo 2**: Identificar un listado de métricas relevantes para caracterizar la actividad en redes sociales.
- **Objetivo 3**: Diseñar una base de datos que para encargarse del almacenamiento de toda la información necesaria, que puede ser de un tamaño considerable.
- **Objetivo 4**: Desarrollar una aplicación web que se encargue de la comunicación con el cliente que la va a consumir, así como de recuperar los datos de la API de la red social, transformarlos y generar un modelo de datos adecuado para guardarlo en la base de datos para su posterior uso.
- **Objetivo 5**: Desarrollar un frontal capaz de mostrar las analíticas que hemos conseguido en el sistema. Que sea entendible y fácil de usar.
- **Objetivo 6**: Preparar el sistema para su despliegue.

1.3. Metodología de trabajo

La metodología elegida para la elaboración del TFG, se ha basado en la metodología ágil *Kanban* sumada a ciertas prácticas de *Scrum*. El desarrollo ágil envuelve un enfoque para la toma de decisiones en los proyectos de software, basados en el desarrollo iterativo e incremental, donde los requisitos y soluciones evolucionan con el tiempo según la necesidad el proyecto [Gon].

Esto ha permitido una estrategia de desarrollo incremental, en lugar de la planificación y ejecución completa de cada una de las diferentes fases de desarrollo de forma secuencial. El objetivo principal es disponer de un producto que enseñar al posible cliente aun en fases tempranas del desarrollo. En la Figura 1.2 se puede observar un simple ejemplo de Kanban.

Para el desarrollo del proyecto se ha apostado por la consecución de las siguientes tareas:

- Evaluación de las APIs de las redes sociales (Twitter, Facebook, Linkedin e Instagram).
- Desarrollo de los componentes esenciales: integración, almacenamiento y visualización de datos.
- Implementación de la lógica de extracción de tuits y posterior almacenamiento.
- Diseño de la pantalla de visualización de métricas.

[5]Application Programming Interface F

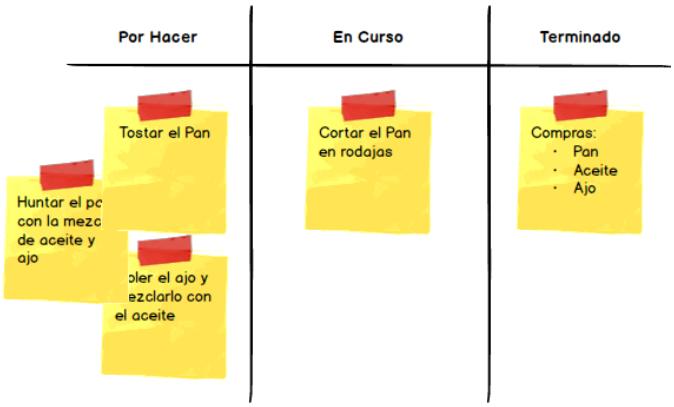

Figura 1.2: Ejemplo Kanban [Tra]

- Generación de algoritmo para la recuperación de comentarios sobre los tuits.
- Evolucionar el diseño de la pantalla de visualización para añadir la información de los comentarios.
- Implementar el sistema de comparación de analíticas.

1.4. Estructura del documento

El documento está compuesto de 6 capítulos y 6 apéndices. El primer capítulo contiene una introducción al TFG. El capítulo 2 analiza las diferentes funcionalidades que ofrecen en la actualidad algunas redes sociales relevantes y la importancia del análisis de sentimiento.

El capítulo 3 describe el diseño de la aplicación, detallando los requisitos, el diseño de la arquitectura y su funcionamiento conceptual, el modelo de datos, la comunicación entre los componentes y las métricas propuestas. El capítulo 4 profundiza en los aspectos de implementación, las tecnologías empleadas, la integración con Twitter y la interfaz donde se muestran las analíticas.

El capítulo 5 muestra dos casos de uso ilustrando cómo funciona el sistema y el valor que aporta. Finalmente, el último capítulo muestra algunas conclusiones y caminos a seguir para su evolución futura.

El apéndice A muestra las horas invertidas en el proyecto, representadas en un diagrama de Gantt. El apéndice B presenta un análisis empresarial de algunas de las herramientas de ámbito similar a la desarrollada. El apéndice C describe la evolución del desarrollo del sistema. El apéndice D muestra las pruebas realizadas al sistema. Por último, el apéndice F contiene un glosario.

Capítulo 2

Estado del Arte

En esta sección se analizan las funcionalidades que ofrecen las diferentes APIs de las redes sociales y se evalúa la importancia del análisis de sentimiento en ellas.

2.1. Análisis de las diferentes APIs

En esta sección se determinará por qué este TFG solo ha enfocado la integración con la API de Twitter, ya que el resto de plataformas presentan limitaciones para su desarrollo. A pesar de este hecho, el sistema quedará preparado para poderse extender a más redes sociales, como las que se comentaron anteriormente. Para ello, se ha hecho un análisis de lo que ofrecen y no ofrecen las redes sociales de interés para nuestro sistema. Aquí se van a analizar las APIs de Twitter, Facebook, Instagram y LinkedIn. Cada una de ellas tiene su propio proceso de autenticación.

Las APIs de estas redes sociales son muy amplias, disponen de muchas posibilidades y métodos para conseguir distintos datos. En estas APIs es fácil conseguir información sobre la propia cuenta, pero conseguirla sobre otras cuentas no es tan sencillo. El problema que se ha tratado de resolver en este análisis es el de conseguir información de cuentas y sus publicaciones que no sean propias y, claro está, que sean abiertas, o si son privadas que tengamos algún tipo de interconexión (por ejemplo, amistad) para poder acceder a los datos como si fuera pública.

Para el uso de todas estas APIs es necesario registrarse como desarrolladores. Normalmente funcionan enlazando una cuenta privada del usuario en la red social, creándose una extensión de la cuenta. Una vez registrado se procederá a crearse una aplicación en las plataformas (la creación en las distintas redes es muy similar, facilitando los campos que te marca como obligatorios), cada una de estas te facilita unas claves de identificación de aplicación para poder usarla en el sistema desarrollado, posteriormente se detallará la creación de una aplicación de Twitter.

Facebook. Esta API es una parte de todo el ecosistema de Facebook para desarrolladores y ofrece muchos servicios para la creación de aplicaciones internas, realidad aumentada, inteligencia artificial, campañas de publicidad, comercio electrónico... Lo que se ha analizado, es el uso de la API que está en una sección llamada "API y SDK". Este API tiene una amplia colección de datos que se pueden solicitar. Se podrían recuperar la mayoría de datos de la propia cuenta de usuario sin problema, el problema es que para los datos que requerimos necesitamos verificación individual o verificación de negocio, esto quiere decir que la cuenta de desarrollador esté verificada por Facebook para que pueda usar su API. Se ha solicitado como uso individual, siendo necesario adjuntar un documento de identificación tal como DNI, carné de conducir o pasaporte.

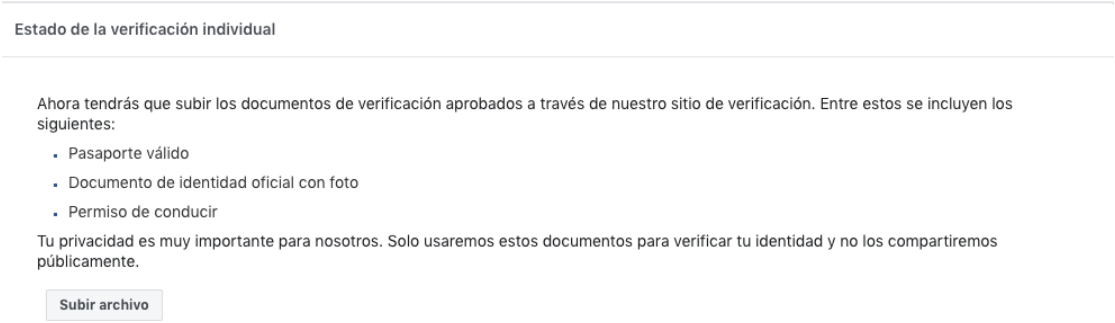

Figura 2.1: Documentación necesaria para verificación individual [Facb]

La verificación individual tarda entre 4 a 5 días una vez presentada la documentación requerida, pero este tipo de verificación no da acceso a todos los datos que ofrece la API. Existe una segunda forma de verificarse, la verificación de negocio. Para ello, no sirve una cuenta de usuario de Facebook normal (se debe disponer de una cuenta de Business Manager en Facebook) y es necesaria documentación referente a la empresa como el CIF. Además, se debe explicar cuál va a ser el uso y mandar un vídeo mostrando cómo el sistema desarrollado utiliza los datos obtenidos. Esto se revisaría por Facebook en un plazo aproximado de 5 días. Por estas razones, la integración de la API de Facebook se considera trabajo futuro.

Figura 2.2: Ejemplo de verificación de funciones de la API de Facebook [Facb]

En esta imagen se puede observar las verificaciones que son necesarias para conseguir el acceso a esta funcionalidad de la API de Facebook.

Instagram. La API de Instagram está integrada dentro de la de Facebook ya que Instagram pertenece a esta compañía. Para recuperar cualquier información, aunque sea de la propia cuenta de usuario, es necesario disponer de una verificación de negocio 2.3 2.4, ya comentada anteriormente. Por esta razón, también se considera trabajo futuro.

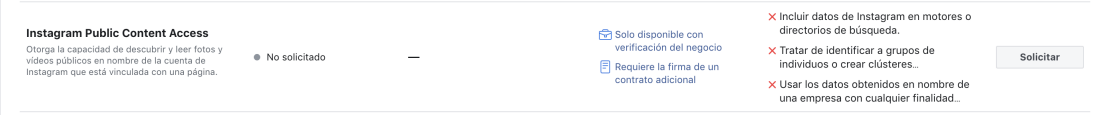

Figura 2.3: Solicitud de acceso público a la API de Instagram [Facb]

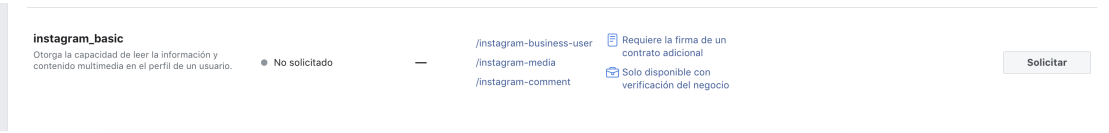

Figura 2.4: Solicitud de acceso básico a la API de Instagram [Facb]

Aunque esté dentro de la API de Facebook, la API de Instagram es mucho más clara. En concreto, en esta todo son usuarios (en Facebook hay usuarios, relaciones, páginas, empresas y cambia el modo de buscar información de un usuario normal o una página de marca o empresa) y la lógica de la red es más sencilla.

Una vez se consiguiera la verificación de negocio y hecha la demostración de uso de como se van a usar los datos, se tendrá acceso a la API de Instagram en la que recuperar los usuarios con estas peticiones sencillas y explicadas detalladamente en su documentación [Faca].

LinkedIn. Para poder crear aplicaciones de LinkedIn es necesario proporcionar información de empresa, por lo que también se considera trabajo futuro.

En la Figura 2.5 se puede observar que para poder finalizar la configuración de la aplicación de LinkedIn es necesario introducir la información de una empresa. Dependiendo de lo que se requiera hará falta pagar, como son las campañas de publicidad o difusión, pero el uso de la API sería gratuito.

A partir de la documentación se intuye que sería posible recuperar la información necesaria, aunque existe una distinción entre la entidad empresa y la entidad usuario. Dependiendo de lo que se quiera analizar, se debería usar un modo de obtener los datos u otro, pero se podría recuperar la información necesaria [Lin].

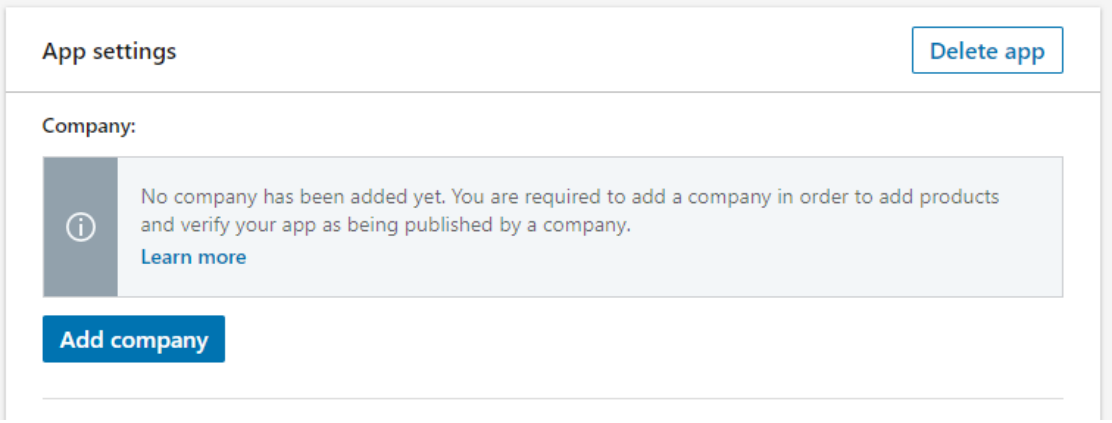

Figura 2.5: Creación aplicación de LinkedIn [Facb]

Twitter. Para crear una aplicación de Twitter basta una cuenta de usuario normal y no es necesaria una cuenta de empresa, por eso hay tantos estudios y uso de esta API, aplicaciones de análisis de sentimiento, aplicaciones de gestión de cuentas, etc.

La API de Twitter es muy extensa pero en este proyecto se van a necesitar cuatro funcionalidades básicas que ofrece:

1. Recuperar un conjunto de usuarios dado un texto. Esto será de utilidad cuando el usuario no sabe el nombre exacto de la cuenta a la que quiere realizar una analítica, pudiendo posteriormente elegir entre un listado de esas cuentas viendo datos relevantes como nombre completo, nombre de usuario o descripción [Twib].

2. Recuperar información detallada de una cuenta concreta enviando como parámetro el nombre de usuario. De aquí se recuperarán los datos generales como descripción, localización, nombre completo..., así como datos relevantes para las métricas que se van a usar, como número de seguidores, seguidos y publicaciones [Twib].

3. Recuperar información sobre las publicaciones/tuits, como número de me gustas o favoritos, tipo de contenido del tuit, menciones o hashtags [Twic].

4. Recuperar comentarios sobre los tuits. Sin embargo, como no existe un método directo de conseguir los comentarios que existen sobre un tuit, se ha tenido que utilizar el método genérico de recuperar tuits que tiene Twitter y se ha tenido que crear un algoritmo para poder recuperarlos [Sea], como se explicará en la Sección 4.1.5.

La API de Twitter tiene ciertas restricciones por tipo de cuenta pero solo afectan a la búsqueda de comentarios sobre tuits, ya que usa un método genérico de obtención de tuits.

2.2. Análisis de sentimiento

El análisis del sentimiento tiene como objetivo el tratamiento computacional de opiniones. Una opinión es una valoración positiva o negativa acerca de un producto, servicio, organización o persona. La llegada de las redes sociales ha catapultado el interés y notoriedad sobre este campo de investigación de la Inteligencia Artificial debido a la importancia que supone saber la valoración que tienen miles de entidades para empresas, organizaciones y consumidores. Este gran volumen de información y el aumento de la potencia de computación de los ordenadores han posibilitado la aplicación de técnicas de aprendizaje automático para la clasificación de textos en base a su polaridad sentimental.

Para cualquier compañía es fundamental poder conocer qué piensan sobre ellas sus potenciales clientes y poder detectar lo antes posible cualquier deterioro en su imagen corporativa y en la manera en la que el público percibe el uso de la marca. Por ello, existen muchos trabajos en la literatura sobre el análisis de sentimiento de datos de Twitter [Sob]; [GBH09]; [Ala17]; [Dra18].

Un ejemplo concreto de aplicación de análisis del sentimiento en redes sociales es predecir el producto necesario que tener disponible en el mercado. Es posible construir una estrategia rentable para operar en el mercado analizando un grupo de tuits, prediciendo el sentimiento predominante de ellos sobre distintas compañías y creando una regla para maximizar las ganancias [Sim].

En este TFG se ha introducido el análisis de sentimiento en los comentarios sobre las publicaciones, lo que ayudará a generar unas métricas más completas. Esto le puede servir al usuario para analizar la opinión que se tiene sobre sus publicaciones a lo largo del tiempo y si tiene que replantearse cambios, o para comparar con otras cuentas en redes sociales. Los trabajos anteriormente citados son complementarios a nuestro sistema y podrían utilizarse en él.

Capítulo 3

Diseño

Este capítulo se centra en el diseño de la arquitectura software, definiendo de forma abstracta, los componentes que intervienen en el sistema, detallando además, los requisitos detectados, las métricas a considerar, el modelo de datos y la comunicación.

3.1. Análisis de requisitos

En el apartado del estado del arte se han comentado ciertas limitaciones y necesidades que presentaba el sistema. Siguiendo la calificación de ingeniería del software, en la Tabla 3.1 se van a especificar los requisitos funcionales, comportamientos específicos del sistema, y los requisitos no funcionales o atributos de calidad.

Requisitos Funcionales	
Nº	**Descripción**
RF1	El sistema permitirá la creación de usuarios.
RF2	Poder autenticarse en Twitter.
RF3	Guardar los tokens de acceso del usuario registrado previamente.
RF4	Tener una lista de cuentas de Twitter del usuario registrado, para su autenticación.
RF5	Recuperar los tuits de un usuario especificado y guardar la información relevante en la base de datos.
RF6	Obtener todas las métricas mencionadas 3.2 y guardarlas en la base de datos.
RF7	Tener un historial de analíticas.
RF8	Recuperar los comentarios de los tuits analizados en la búsqueda de la cuenta de usuario.
RF9	Poder recuperar sugerencias de usuarios a los que realizar el análisis dada una cadena de texto y mostrarlas por pantalla.
RF10	Mostrar en una web las analíticas de una búsqueda.
RF11	Mostrar por pantalla una lista con el historial de búsquedas realizadas.
RF12	Poder seleccionar una búsqueda del historial para mostrar las analíticas de la misma.
RF13	Poder hacer una comparación entre búsquedas de una cuenta de usuario entre dos puntos de tiempo distintos.

RF14	Poder hacer una comparación entre búsquedas de distintas cuentas de usuario.

Tabla 3.1: Requisitos Funcionales

Requisitos No Funcionales	
Nº	**Descripción**
RNF1	El sistema funcionará en Google Chrome
RNF2	El sistema extraerá los datos de la API de Twitter
RNF3	Se puede abrir en dispositivos móviles
RNF4	El historial de analíticas no deberá incluir el nombre de usuario si es distinto al usuario actual por privacidad.
RNF5	Utilizar la base de datos para no limitarse a los 7 últimos días

Tabla 3.2: Requisitos No Funcionales

Como se puede observar en estas listas de requisitos, el sistema requiere ciertas funcionalidades que no ofrecen los sistemas comentados en el capítulo anterior.

No se han añadido requisitos funcionales de integración sobre las otras redes sociales ya que no se ha contemplado para este TFG, quedando como trabajo futuro.

3.2. Métricas

Estas métricas están basadas en un sistema de categorías de comunicación propuesto por Borja Aso [Aso19].

Primero, consideramos métricas sobre el usuario fáciles de conseguir y que se pueden ver a simple vista accediendo a la cuenta de Twitter concreta:

1. Seguidores (*followers*), o número de usuarios que siguen a ese usuario.
2. Seguidos (*following*), o número de usuarios que sigue ese usuario.
3. Número de tuits o publicaciones realizados por ese usuario.

Las siguientes métricas que vamos a considerar van a ser sobre cada uno de los tuits de ese usuario. Estas no se ven a simple vista al acceder a la cuenta del usuario de Twitter, deberías recorrer todos los tuits y obtener toda la información relevante de cada uno,

4. Tuits analizados, ya que en muchos casos no se va a poder analizar la totalidad de ellos, y rango de fechas que abarcan.

5. Número de favoritos (*favourites*) o me gustas (*likes*) dados por otros usuarios a todos los tuits del usuario.

6. Retuits, o número total de reenvíos de tuits que han hecho otros usuarios a todos los tuits del usuario.

7. Comentarios (*replies*), o número total de respuestas de otros usuarios a todos los tuits del usuario.

8. Mentions, o número total de menciones realizadas por el usuario de la búsqueda realizada en el total de sus tuits.

9. Hashtags, o número total de hashtags escrito por el usuario de la búsqueda realizada en el total de sus tuits.

10. Media, o número total de archivos media en el total de los tuits.

11. URL, o número de tuits con una o más URLs.

12. Frecuencia de hashtags (*hashtags frequency*) o número de repeticiones de cada uno de los hashtags escritos en el total de los tuits, lo que sirve para saber cuales son los más y menos usados. Se mostrará en formato de nube de palabras (*tag cloud*[1]).

13. Frecuencia de las menciones (*mentions frequency*) o número de repeticiones de cada uno de las menciones escritas en el total de los tuits, lo que sirve para saber cuales son las más usadas. Se mostrará en formato de nube de palabras (*tag cloud*).

14. Tuits por día (se mostrará en formato de gráfica) o frecuencia total de tuits al día.

15. Tuits con solo texto por día (se mostrará en formato de gráfica) o frecuencia de tuits con solo texto al día.

16. Tuits con texto y URL (se mostrará en formato de gráfica) o frecuencia de tuits con texto y URL al día.

17. Tuits con texto e imagen (se mostrará en formato de gráfica) o frecuencia de tuits con imagen y texto al día.

18. Tuits con texto e imagen y URL (se mostrará en formato de gráfica) o frecuencia de tuits con imagen, texto y URL al día.

19. Tuits con solo imagen (se mostrará en formato de gráfica) o frecuencia de tuits con solo imagen al día.

[1] **https://www.humanlevel.com/diccionario-marketing-online/nube-de-tags**

20. Tuits con texto y vídeo (se mostrará en formato de gráfica) o frecuencia de tuits con vídeo y texto al día.
21. Tuits con texto e vídeo y URL (se mostrará en formato de gráfica) o frecuencia de tuits con vídeo, texto y URL al día.
22. Tuits con solo vídeo (se mostrará en formato de gráfica) o frecuencia de tuits con solo vídeo al día.
23. Total de tuits con solo texto.
24. Total de tuits con texto y URL.
25. Total de tuits con texto e imagen.
26. Total de tuits con texto, imagen y URL.
27. Total de tuits con solo imagen.
28. Total de tuits con texto y vídeo.
29. Total de tuits con texto, vídeo y URL
30. Total de tuits con solo vídeo.

En la API de Twitter existen ciertas restricciones con los tuits: se pueden conseguir tuits de cualquier fecha de un usuario pero con un máximo de los 3200 últimos tuits realizados. Por ejemplo, de una cuenta que fue creada hace 10 años y que ha tenido poca actividad en las redes sociales (1500 tuits) se van a poder recuperar todos los tuits, pero en el caso de una cuenta que tenga una alta actividad en las redes sociales (digamos 70000 tuits), solo se podrán recuperar como máximo las 3200 publicaciones más recientes.

Las siguientes métricas son sobre los comentarios. La API de Twitter tiene algunas limitaciones importantes sobre los comentarios, al no tener ninguna forma directa de poder recuperar todos: solo se pueden recuperar hasta 7 días antes de la búsqueda. Para solucionarlo, se ha tenido que desarrollar un algoritmo para poder recuperar el mayor número posible de los mismos, que se explicará en la Sección 4.1.5. En una gráfica se mostrarán todos los tuits que tengan comentarios asociados y se mostrarán los siguientes datos:

31. Valoración (*score*) o número que estima la polaridad de un tuit. Un valor mayor que 0 indica polaridad positiva, un valor menor que 0 indica polaridad negativa y un valor igual a 0 indica polaridad neutra.
32. Número de comentarios positivos por cada tuit.
33. Número de comentarios negativos por cada tuit.

34. Número de comentarios neutrales por cada tuit.
35. Número total de comentarios.
36. Media de comentarios recibidos, total de comentarios entre tuits analizados.
37. Tuits con comentarios, número de tuits analizados de los 7 días antes de la búsqueda.
38. Número total de comentarios positivos.
39. Número total de comentarios negativos.
40. Número total de comentarios neutrales.
41. Número de tuits con valoración positiva.
42. Número de tuits con valoración neutral.
43. Número de tuits con valoración negativa.

Además de las limitaciones comentadas anteriormente, Twitter limita el número de peticiones que se pueden realizar sobre sus recursos, el método de obtención de comentarios son 180 peticiones por usuario y 450 por aplicación cada 15 minutos [Sea] y la obtención de tuits serían 900 tanto usuario como aplicación cada 15 minutos [Twic].

3.3. Diseño arquitectural

En esta sección se va a comentar la arquitectura que se sigue, así como los componentes involucrados.

Se ha diseñado una arquitectura de 2 capas, que se usa para describir sistemas cliente / servidor y se añade también una fuente de datos externa que es la API de Twitter con la que también se mantendrá la interacción (ver Figura 3.1).

En un vistazo general, los componentes diseñados y sus relaciones se explican de la siguiente forma. De iniciar el flujo se encargará el cliente ya que es el que requerirá todos los datos del servidor para visualizarlos en la pantalla. A su vez, el servidor se comunicará directamente con la base de datos y con la API.

- **Cliente**. Nuestro cliente Web se encargará de toda la visualización del sistema, será el encargado de comunicarse con el servidor, recuperar datos y mostrarlos. Tiene cierta lógica como modificación de colecciones pero en lo que se centra es en recuperar la información, mostrarla como es debido y controlar el flujo de interacciones con el usuario.
- **Servidor**. Tendrá 3 funciones:

1. Comunicación con el cliente para el intercambio de datos.
2. Comunicación con la API de Twitter, para la extracción de los datos que requerimos de Twitter.
3. Comunicación con la base de datos para almacenar y recuperar los datos relevantes.

- **Base de datos**. Se encarga de guardar tanto la información de nuestros usuarios como los datos relevantes que obtenemos de Twitter tales como tuits, analíticas de los tuits e información de cuentas de Twitter, como veremos en la Sección 3.4

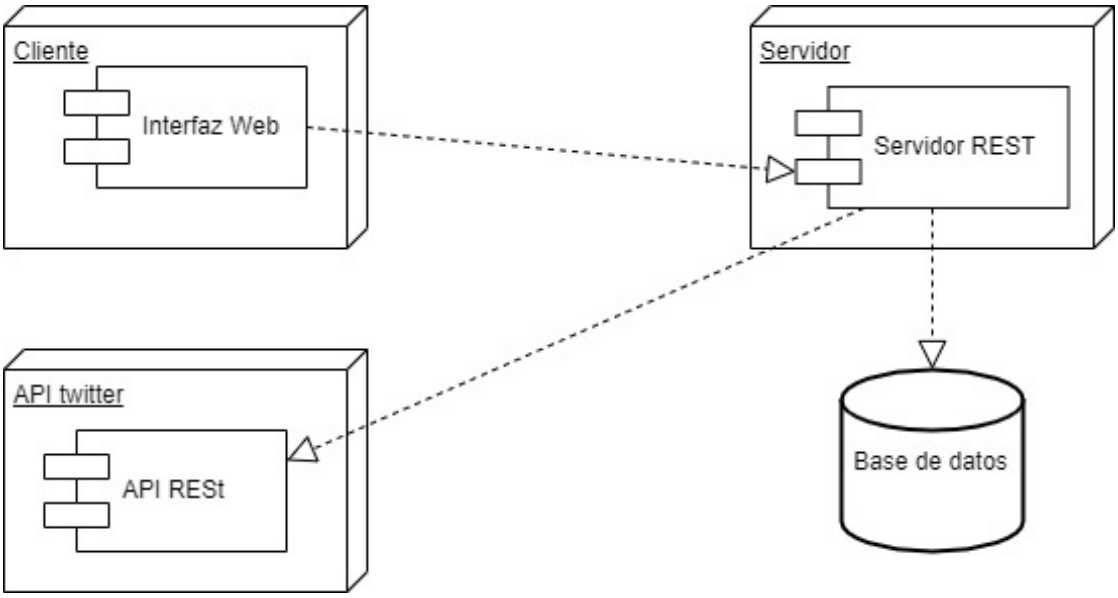

Figura 3.1: Arquitectura del sistema

Gracias a que estas capas están desacopladas se ha ganado en flexibilidad e independencia, pudiendo usar cada una de ellas una tecnología diferente siempre y cuando se mantenga el formato de intercambio de datos.

Para la escalabilidad y extensión de este sistema, esta flexibilidad, es de suma importancia ya que se podría crear una aplicación para un dispositivo móvil independiente del cliente web respetando el formato de intercambio de datos. Así mismo, podrían convivir las dos aplicaciones clientes.

En la parte del servidor se podría ampliar el sistema replicando esta parte. Así, dependiendo de la carga se podría balancear y conseguir un sistema de alta disponibilidad, teniendo tiempos cortos de espera y evitando la caída total del sistema si uno de los servidores dejara de funcionar, pues podría tener varios de reserva.

Al ser un sistema desacoplado, se podrían crear otros sistemas servidor en cualquier otra tecnología cómo Java, para ampliar la funcionalidad con otras redes sociales.

La base de datos es independiente del servidor pero sigue teniendo un fuerte acoplamiento, así que si se quisiera cambiar la base de datos a una tecnología diferente, sería necesario modificar gran parte del código. Una posible mejora a futuro sería crear un servidor independiente para la base de datos, un servidor de datos.

3.4. Modelo de datos

El modelo de datos es una de las partes principales del sistema creado, ya que es el almacén de toda la información. Se ha elegido MongoDB, una base de datos NoSQL, para favorecer su uso en máquinas con pocos recursos, adaptándose perfectamente a las necesidades del sistema [Aul].

Este tipo de bases de datos también es muy potente a la hora de escalar, ya que podremos ir replicando la base de datos o distribuyendo en varias máquinas pequeñas sin necesidad de tener solo una máquina de alto rendimiento y bastante más cara.

Otra característica importante para nuestro sistema es que está muy bien optimizado a la hora de manejar grandes cantidades de datos, ya que nuestras tablas de tuits pueden alcanzar volúmenes de datos por encima de 3200 registros y se quiere recuperar los datos que necesitemos en un tiempo óptimo.

El modelo de datos está separado en 4 colecciones de datos no relacionadas entre ellas:

1. **Users** tiene los datos de los usuarios, información básica y cuentas de Twitter que tienen vinculada cada usuario y sus datos de acceso al API (ver Figura 3.2).

_id	ObjectId("5d9870d4e8c7c90cc457cf74")	ObjectId
b_borrado	false	Bool
email	ruben.gabas@gmail.com	String
password	4MkDWJjdUvxlxBRUzsnE0mEb+zc=	String
cuentas	Array[1]	Array
__v	2	Int32
(2) ObjectId("5d937cb3d04900455c529113")	{ 6 attributes }	Document

Figura 3.2: Información usuarios en la base de datos

2. **Analitycs** guarda la información básica de la analítica (ver Figura 3.3), como información directamente vinculada a la cuenta de usuario, ya que esta es muy rápida de conseguir y se puede devolver rápidamente al consumidor de los datos no teniéndolo mucho rato en espera. Por ejemplo,el nombre de usuario de la cuenta de Twitter que se está analizando, en qué fecha y hora se hizo la búsqueda ...

3. **InfoAnalitycs** guarda las métricas fruto del análisis de la cuenta especificada (ver Figura 3.4). Cada registro de Analitycs genera un registro en esta colección.

4. **Tuits** guarda la lista entera de tuits por nombre de cuenta de usuario de Twitter analizada (ver Figura 3.5).

tweets	[ ]	Array
id	480,382,619 (0.48G)	Int32
id_str	480382619	String
user_searcher	maria	String
type	twitter	String
name	Real Zaragoza	String
screen_name	RealZaragoza	String
location	Zaragoza	String
description	⚽ Cuenta oficial del Real Zaragoza \n💙🤩 Categorías inferiores: @RZcantera	String
url	https://t.co/vW0Sipu1kG	String
followers_count	288,009 (0.29M)	Int32
friends_count	91	Int32
listed_count	1,360 (1.4K)	Int32
statuses_count	56,173 (56.2K)	Int32
profile_background_image_url	http://abs.twimg.com/images/themes/theme1/bg.png	String
profile_image_url	http://pbs.twimg.com/profile_images/1186038126017429504/beFw9mys_normal.jpg	String
date_of_search	13/11/2019, 18:49:55	Date
__v	0	Int32

Figura 3.3: Información analíticas en la base de datos

_id	ObjectId("5dcc4243f2c4040d999805	ObjectId
userMentions	Array[1529]	Array
hashtags	Array[2239]	Array
postsInDay	Array[120]	Array
postsInMonth	Array[4]	Array
userMentionsGrouped	Array[561]	Array
hashtagsGrouped	Array[155]	Array
replies	Array[79]	Array
id_of_analityc	5dcc4243f2c4040d99980570	String
state	Done	String
dateInit	16/07/2019, 23:10:42	Date
dateEnd	13/11/2019, 18:10:03	Date
__v	1	Int32
favoritesTotal	318,579 (0.32M)	Int32
hashtagsTotal	2,239 (2.2K)	Int32
mediasTotal	882	Int32
ownPosts	3,016 (3.0K)	Int32
retweetsTotal	79,277 (79.3K)	Int32
screen_name	RealZaragoza	String
sharePosts	214	Int32
urlsTotal	2,109 (2.1K)	Int32
userMentionsTotal	1,529 (1.5K)	Int32

Figura 3.4: Información del detalle de las analíticas en la base de datos

_id	ObjectId("5dcc42e5f2c4040d999805f4")	ObjectId
tweets	Array[3016]	Array
id_of_analityc	5dcc4243f2c4040d99980570	String
screen_name	RealZaragoza	String
__v	0	Int32

Figura 3.5: Información sobre tuits en la base de datos

Cada creación de usuario en la aplicación generará un registro en la colección de usuarios, así como cada petición de autenticación con Twitter modificará el registro del usuario en la base de datos añadiendo información (tokens de autenticación).

Pese a las limitaciones de Twitter de recuperar un máximo de 3200 tuits por búsqueda de una misma cuenta y los últimos 7 días de comentarios/replies sobre los tuits, el modelo de datos permite almacenar los datos para disponer de más de 3200 tuits y más de 7 días desde la ejecución del análisis, ya que se dispondrá de un histórico de datos desde la primera vez que se hizo la búsqueda de la cuenta en cuestión.

3.5. Comunicación

En esta sección se va a hablar de la comunicación entre componentes, sus interacciones y se han descrito tres diagramas de secuencia para registro, inicio de sesión y generación de analíticas.

Los componentes que forman el sistema se han comentado en el diseño arquitectural (Sección 3.3); aquí se comentará cual es la tecnología de cada uno ellos y qué protocolos de comunicación usan.

3.5.1. Elementos tecnológicos

Disponemos de tres componentes tecnológicos internos al sistema, que son: un cliente que ha sido implementado en React.js, un servidor implementado en Node.js y Express.js, y la base de datos en MongoDB. Este conjunto tecnológico se denomina Mern (Mongo, Express, React y Node) y está basado en Javascript [Sam]. Además, como componente externo se utilizará la API de Twitter.

Como formato de intercambio de datos se usará JSON, ya que es la estructura de datos natural que se usa en Javascript. La comunicación entre cliente y servidor será vía REST. [Rib]

3.5.2. Secuencia de registro

En la Figura 3.6 se puede observar la comunicación entre los componentes: cliente, servidor y base de datos. El componente externo API de Twitter no se muestra, ya que en este momento no es necesario. Se puede observar que hay dos operaciones en el cliente que inician la secuencia, esto es porque una vez que se ha creado el usuario y se han recuperado los tokens de autorización del sistema, se lanza otra operación para comprobar que se tiene acceso a la parte interna del sistema, que requiere los tokens.

La comunicación siempre tiene la misma estructura: el cliente se comunica con el servidor, el servidor se comunica con el cliente y la base de datos, y la base de datos con el servidor. Toda la lógica de la aplicación se encuentra en el servidor.

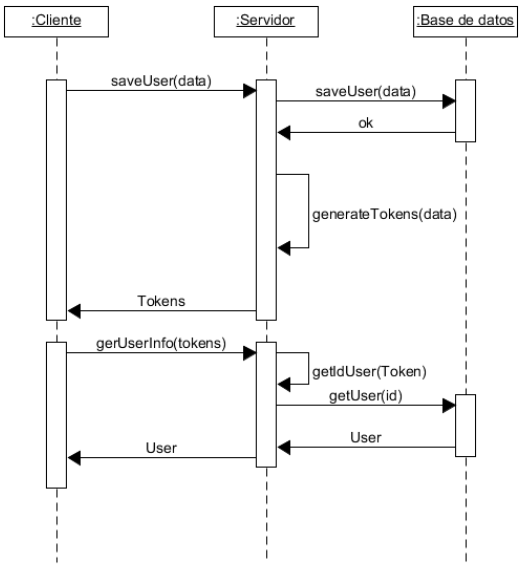

Figura 3.6: Diagrama de secuencia de registro

3.5.3. Secuencia de inicio de sesión

La Figura 3.7 muestra la comunicación entre componentes necesaria para el inicio de sesión satisfactorio en el sistema. Igual que en la secuencia anterior, solo son necesarios los componentes internos del sistema y no la API de Twitter.

La composición de la secuencia sería similar, con dos operaciones: la primera se encarga de iniciar la sesión y la segunda se encarga de comprobar que se tiene acceso a la parte interna del sistema, que requiere los tokens de autorización.

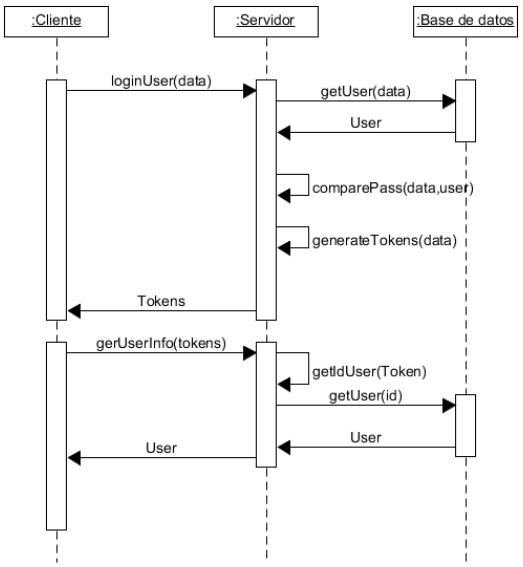

Figura 3.7: Diagrama de secuencia de inicio de sesión

3.5.4. Secuencia de generación de analíticas

La siguiente imagen muestra la comunicación entre componentes necesaria para la generación de analíticas en el sistema. En esta secuencia se introduce el componente Twitter API además de los 3 del sistema desarrollado.

Esta secuencia tiene una lógica más compleja que las dos anteriores. El flujo empezaría con la petición del cliente para empezar a generar analíticas y el servidor ejecutaría tres operaciones: la recuperación de información de una cuenta de Twitter, otra de recuperación de tuits de Twitter y la recuperación de los comentarios sobre los tuits.

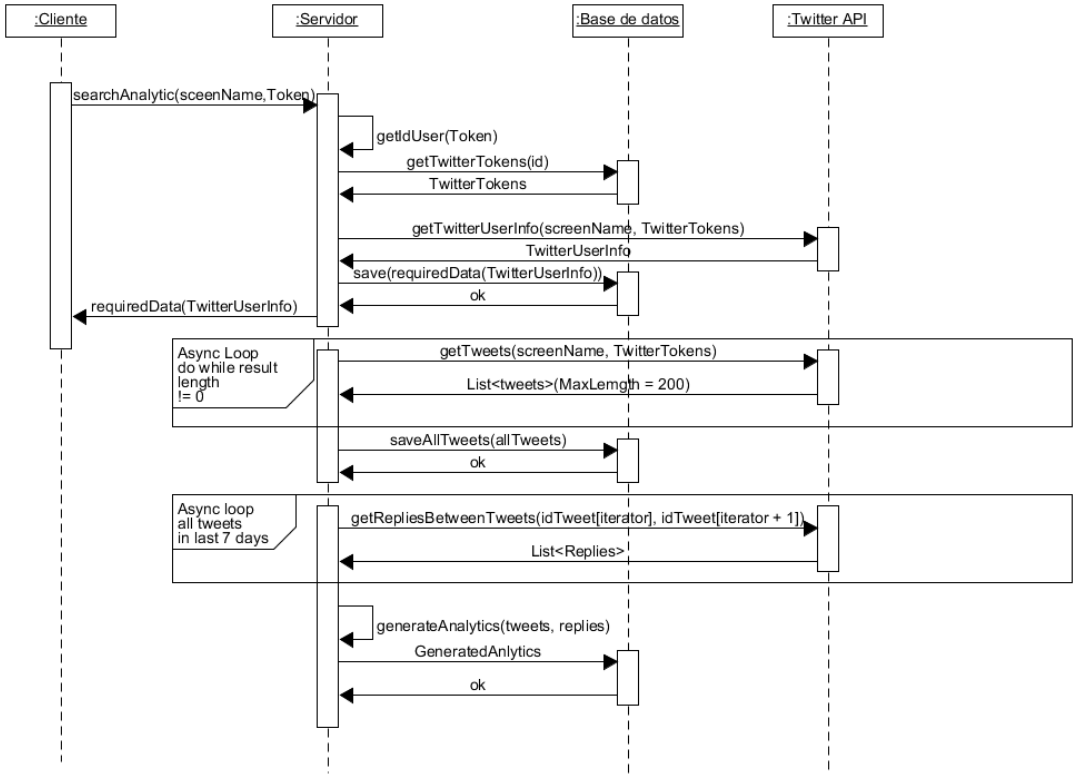

Figura 3.8: Diagrama de secuencia de generación de analíticas

Capítulo 4

Implementación

Este capítulo se va a centrar en la implementación del sistema, su integración con Twitter, las tecnologías, explicación de la interfaz de analíticas y la evolución que ha tenido el sistema.

4.1. Integración con Twitter

En esta sección vamos a hablar de las distintas interacciones de recuperación de datos que tenemos con Twitter, lo que constituye la parte principal del TFG. De la extracción, transformación y almacenamiento de datos de Twitter se encarga el servidor. A continuación se explicarán todas las interacciones con Twitter.

4.1.1. Autenticación con Twitter

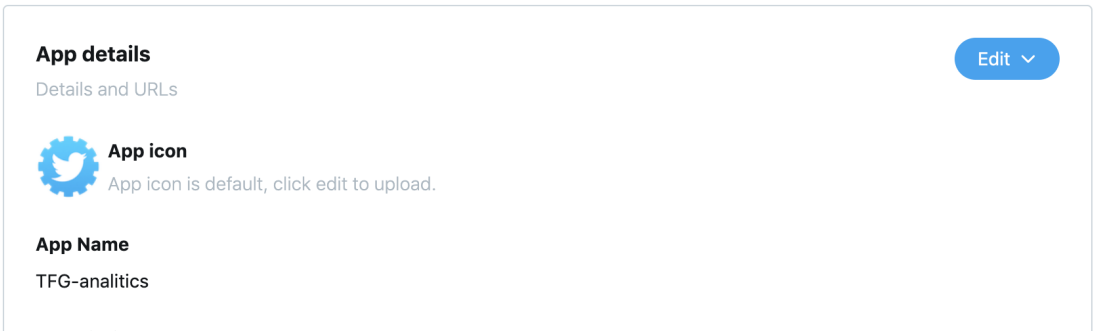

Figura 4.1: Ejemplo de aplicación creada en Twitter

Una vez que se ha creado una aplicación en Twitter (Figura 4.1), en la pestaña de la app de Twitter "Keys and tokens" aparecerán las "consumer API keys" que constarán de dos claves, "API key" y "API secret key", necesarias para insertarlas en la configuración

del sistema que se ha creado y que se pueda autenticar con Twitter. Para la autenticación con Twitter se va a usar Oauth 2 [Twia]. El flujo consta de 3 pasos:

1. El sistema creado redirige a la aplicación de Twitter creada anteriormente.
2. En la pantalla mostrada en la Figura4.2 es necesario iniciar sesión en Twitter, y una vez se hayan validado los datos volverá a redirigir al sistema creado para este TFG.

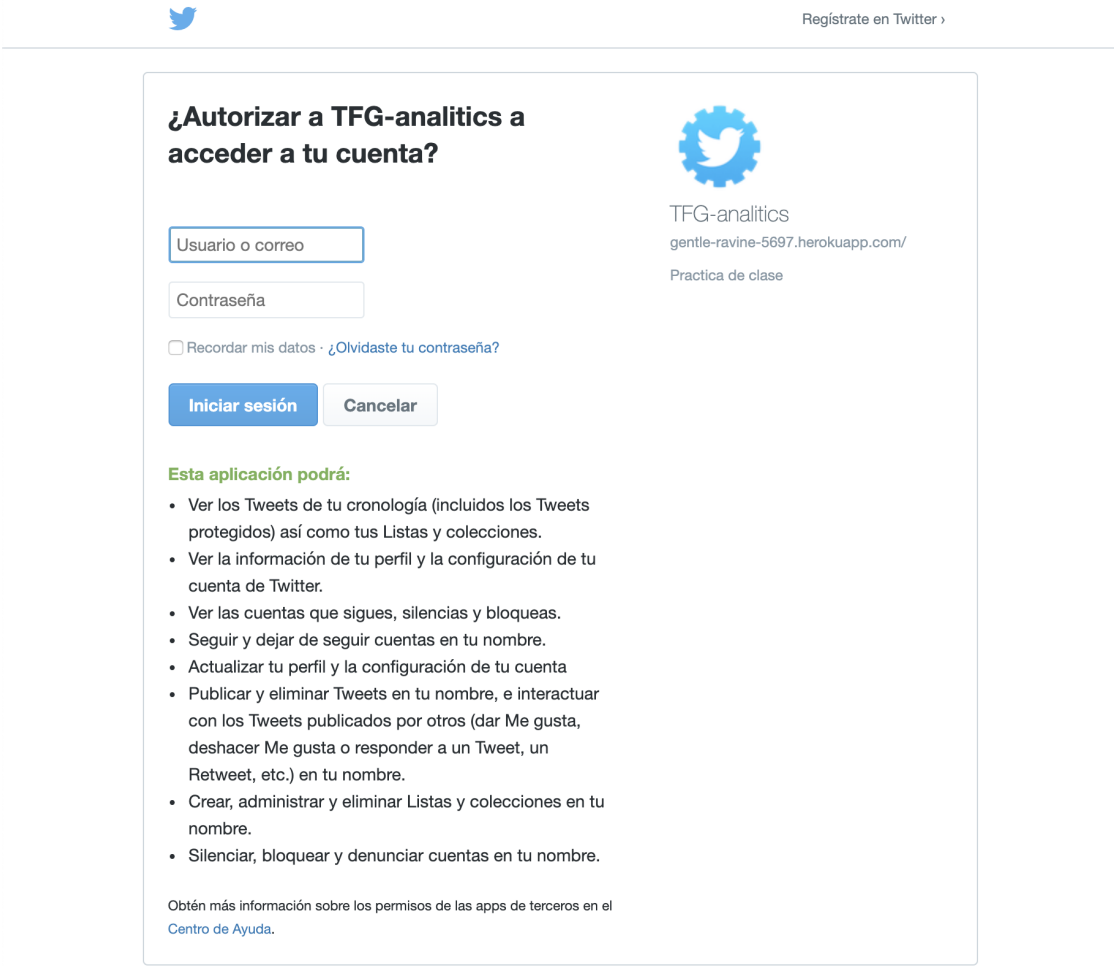

Figura 4.2: Registro a la aplicación de Twitter

3. En la redirección se reciben el “access-token” y el “secret-token” del usuario que ha iniciado sesión a la aplicación de Twitter. Estos “tokens” proporcionan acceso a la API de Twitter y se guardan en la base de datos asociados al usuario del sistema que inició el flujo, para su posterior uso 4.3.

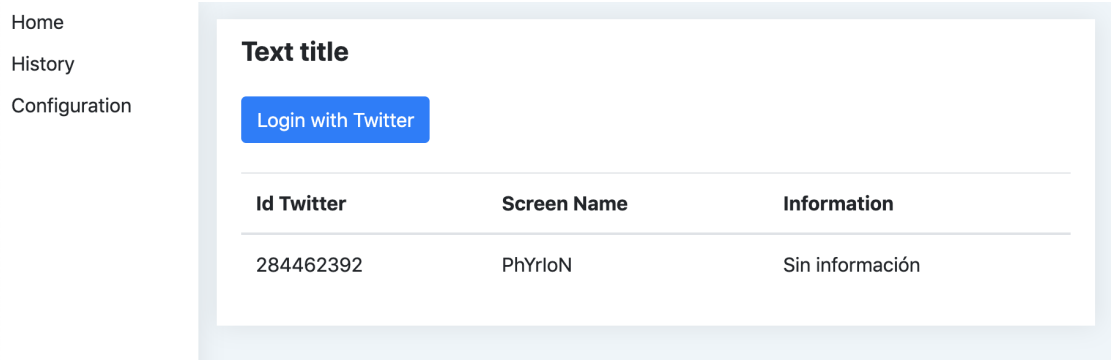

Figura 4.3: Pantalla de autenticación en Twitter

Gracias a este proceso ya se tendrían unos Tokens de autenticación de Twitter asociados a un usuario del sistema que se ha creado.

4.1.2. Recuperación de sugerencias en Twitter

Esta funcionalidad trata de enriquecer la búsqueda, ya que no es necesario facilitar el nombre exacto de la cuenta de usuario a la que realizar el análisis de sus datos. El funcionamiento sería el siguiente: un usuario del sistema introduce un texto al que realizar la búsqueda y el sistema realizará una petición usando la API de Twitter, recuperando una lista de usuarios que, en su nombre completo o nombre de usuario, incluyen el texto especificado. Esto es muy útil si no se sabe exactamente cual es el nombre de usuario de la cuenta de Twitter que se desea analizar.

Como se puede observar en la Figura 4.4, en la aplicación se ha introducido un texto sobre el que realizar la analítica y bajo ello se muestra una lista de posibles usuarios que encajan con ese texto. Cada usuario se compone de: nombre completo, nombre de usuario, descripción e información de la cuenta.

Para realizar la analítica se debe pulsar sobre uno de los usuarios facilitados. También se puede realizar la analítica directamente sin necesidad de recuperar las sugerencias, introduciendo en el campo de texto una @ seguida del nombre de la cuenta de usuario.

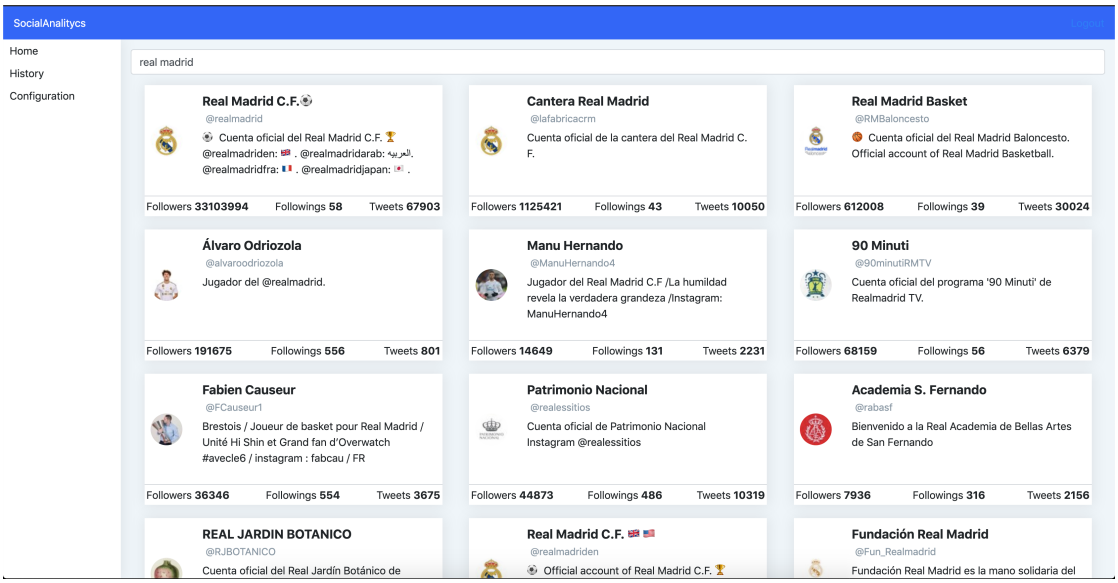

Figura 4.4: Pantalla de sugerencias

4.1.3. Generación de analíticas - Información del perfil

Dado un nombre de usuario de Twitter, el sistema ejecutará el análisis de dicha cuenta de Twitter y el primer paso sería recuperar la información básica de la cuenta, que como datos más relevantes tiene: nombre completo, nombre de usuario, descripción, seguidores y/o seguidos y número total de publicaciones realizadas. En la base de datos se guarda un registro con esta información para su posterior uso.

Como este proceso es rápido, esta funcionalidad es síncrona: en cuanto se recuperan estos datos y se guardan en la base de datos el flujo retorna al cliente del sistema con los datos extraídos de la cuenta.

4.1.4. Generación de analíticas - analíticas de los tuits

Esta funcionalidad se inicia a la vez que la anteriormente comentada, pero con la diferencia de que funciona de forma asíncrona, aunque la otra finalice, esta funcionalidad seguirá su flujo, ya que es una tarea más costosa en tiempo.

La recuperación de tuits tiene limitaciones como que solo se puedan recuperar un máximo de 3200 tuits de la misma cuenta de usuario, los anteriores a ese límite ya son imposibles de recuperar aunque gracias a nuestra base de datos se podrán recuperar si se han realizado analíticas anteriores de esa cuenta de usuario.

El algoritmo de búsqueda trata de recuperar todos los tuits. En cada "petición" el máximo de datos está limitado a 200. Se pedirían estos 200 primeros datos que, por

defecto, van desde el tuit más actual al más antiguo. Para llegar al máximo de 3200 tuits que podemos recuperar tendríamos que hacer esa petición 16 veces, pero si hiciéramos esas 16 peticiones iguales nos devolvería siempre los mismos 200 tuits.

Se solventará este problema con uno de los parámetros que podemos facilitar a la petición de Twitter, "max_id" que lo que específica, es que se busquen tuits con "ids" con un valor numérico más bajo que el que se pasa por parámetro. Se hará la petición de recuperar los 200 tuits iterativamente y en cada iteración se recuperará el tuit más antiguo, al cual se extraerá su id y la siguiente petición se usará como 'max_id' llegando a conseguir todos los tuits y no los mismos 200 repetidamente. Evidentemente no todos los usuarios de Twitter tienen más de 3200, en ese caso solo haría falta realizar las iteraciones necesarias para recuperar el total de sus tuits. [Twic]

Una vez recuperados los tuits se iniciará el cálculo de las analíticas. Existen los siguientes casos: que sea la primera analítica de esta cuenta de usuario, que no sea la primera analítica y tenga menos de 3200 tuits y que no sea la primera analítica y que tenga más de 3200 tuits.

- **Primera analítica de esta cuenta de usuario**. Es el caso más sencillo ya que no tenemos datos de analíticas ni tuits guardados en la base de datos, entonces todos los tuits que recuperemos son válidos y habrá que calcular las 'métricas' y generar el análisis sobre toda la lista de tuits recuperados, ya sea el máximo de los 3200 tuits o menos. Una vez finalice el cálculo de las métricas, se guardarán los datos requeridos de los todos los tuits en la tabla "Tweets" de la base de datos de Mongo con un identificador del nombre de usuario de la cuenta de Twitter al que pertenecen esos tuits para su posterior uso.

- **No sea la primera analítica de esta cuenta y tenga menos de 3200 tuits**. En este caso ya tenemos información de una analítica anterior. Igualmente tenemos que recuperar todos los tuits, ya que se han podido hacer nuevas interacciones sobre ellos (ej: número de me gustas, número de favoritos...) y además puede haber nuevos tuits. Se realizará el cálculo de todos los tuits obtenidos y se guardarán como un nuevo registro en la base de datos de analíticas y los tuits se sobrescribirán con los nuevos.

- **No sea la primera analítica de esta cuenta y tenga más de 3200 tuits**. Ese caso es el más diferente. Primero se recuperan todos los tuits de Twitter posibles, se busca la "id" del más antiguo y se recuperaran todos los tuits de la base de datos de la cuenta de usuario sobre la que se está haciendo la analítica y que tengan 'id' menores que este, ya que estos no han podido ser recuperados de Twitter. En este caso se hace la analítica de la unión de las 2 colecciones de tuits, ya que los tuits recuperados de Twitter pueden tener datos actualizados además de nuevas publicaciones. Además se actualizarán los tuits en la base de datos con la colección completa obtenida de las 2 colecciones anteriormente comentadas.

Gracias a nuestra base de datos, el sistema no estará limitado a 3200 tuits: si se

realizó una primera búsqueda de una cuenta con más de 3200 tuits ese sería el máximo, pero las siguientes búsquedas añadirán nuevos tuits a esos resultados previos.

4.1.5. Generación de analíticas - analíticas de los replies/comentarios

Cuando se habla de replies se hace referencia a los comentarios recibidos sobre los tuits, pero el proceso para recuperarlos no es tan sencillo como el de los tuits, pues la API de Twitter no tiene un método para conseguir todos los comentarios de un tuit especificado.

Se ha partido de un método de búsqueda general de tuits que ofrece Twitter, pero surge el problema de que se está usando una cuenta estándar, gratuita, con un límite de recuperación de datos de solo hasta 7 días antes de ejecutar la búsqueda. Si se usara una cuenta de pago, se podrían llegar a recuperar de 30 días antes y con una cuenta "enterprise", de cualquier fecha.

Como esta búsqueda no es una búsqueda especifica para recuperar comentarios sobre tuits, se ha tenido que elaborar a partir de ella un algoritmo para conseguir una colección de comentarios lo más amplia posible. Una búsqueda básica recuperaría los últimos tuits publicados en todo el sistema de Twitter con un limite por defecto de solo 15 tuits. A ello podemos añadir varios parámetros: "to:@[cuenta_usuario]", que indica que sean tuits con mención a la cuenta que estamos analizando, y un límite de 100 (número máximo admitido). Con esto se tendrían hasta 100 tuits recuperados con mención a la cuenta a la que se está haciendo la analítica, que podrían ser menciones directamente a la cuenta o a cualquiera de sus tuits publicados. Sería un dato poco relevante ya que podría haber tuits con solo un comentario o incluso que todos esos comentarios fueran menciones directamente a la cuenta.

Lo que se va a hacer es una recuperación iterativa, basada en el bloque de tuits recuperados en la Sección 4.1.4, y de estos se recuperarán los que hayan sido publicado en los últimos 7 días. Se va a hacer referencia a ese conjunto de datos como *Lista A*. La petición genérica de tuits usará además de los parámetros comentados anteriormente "max_id" y "since_id". El algoritmo funcionaría de la siguiente manera, como ilustra la Figura 4.5:

1. Se recorre *Lista A* desde el "id" más bajo al "id más" alto.
2. Para cada elemento de la *Lista A* se usa el "id" propio como "since_id" y el "id" del elemento siguiente como "max_id". Se ejecuta la petición general de tuits, lo que recupera una colección de comentarios con "ids" entre "max_id" y "since_id", en adelante nos referiremos a esta lista como *Comentarios_parciales A*. Gracias a esto, no se limitará a la recuperación de 100 tuits.
3. Para cada uno de los comentarios en cada una de las colecciones *Comentarios_parciales A*, se busca el "id" del tuit del que es respuesta y se asocia si coincide con el "id" de alguno de los tuits de la *Lista A*, sino se desecha.

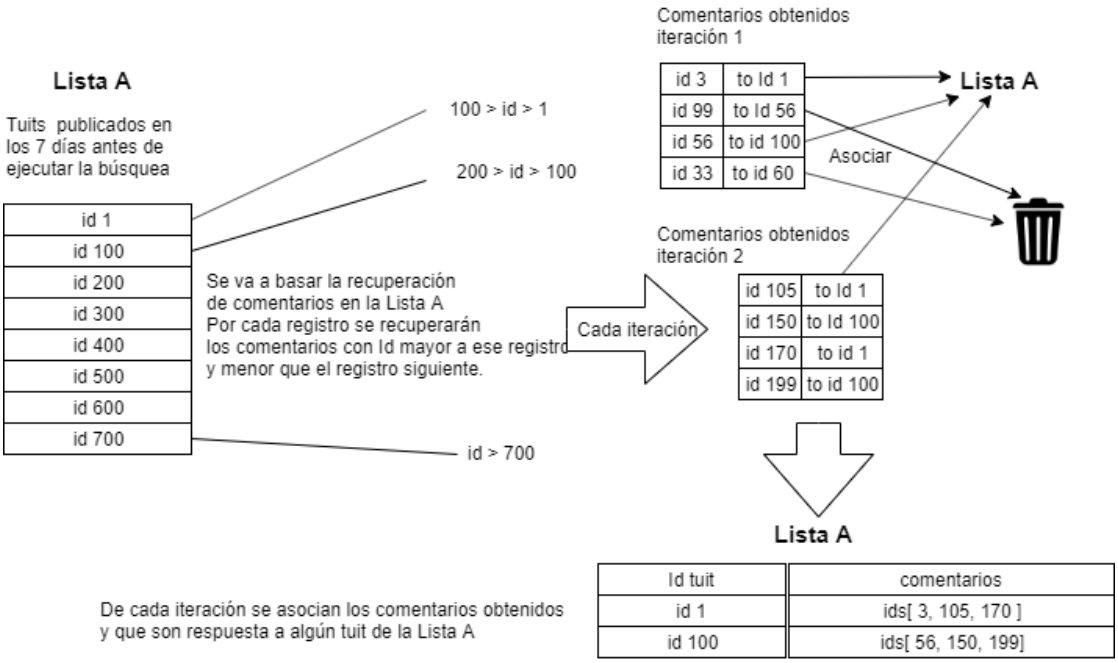

Figura 4.5: Algoritmo de obtención de comentarios

Por cada uno de los comentarios recuperados se realizará un análisis de sentimiento, para lo que se ha utilizado multilang-sentiment[Bar]. Al ser multilenguaje, antes de realizar el análisis de sentimiento se tiene que recuperar el idioma en el que esta escrito el tuit, para eso los comentarios recuperados tienen el campo "lang". multilang-sentiment tiene una lista de idiomas soportados, como hay lenguajes que son soportados por Twitter y no por nuestra biblioteca se va a trabajar con textos en español o inglés, y los que estén en otro idioma se tratarán como comentarios neutros.

Una vez obtenido el lenguaje se podrá proceder a realizar el análisis de sentimiento. Una vez realizado, se obtendría una puntuación o *score* mayor que 0 si es un comentario positivo, menor que 0 si es negativo o 0 si es neutro. Una vez obtenido este valor, guardamos el objeto en la base de datos.

Cada registro de la colección "replies" sería uno de los tuits que se han recuperado de los 7 días anteriores a la búsqueda y tendría el siguiente formato (ilustrado en la Figura 4.6):

- **id** del tuit sobre el que se han analizado sus comentarios.
- **replies** o número de comentarios que se han analizado para ese tuit.
- **score** o valor obtenido como la suma de todos los valoraciones de los comentarios analizados.
- **positive** o número de comentarios con valor positivo para este tuit.

Key	Value	Type
replies	Array[79]	Array
0	{ 6 attributes }	Object
id	1194555006424354800	String
replies	9	Int32
score	3	Int32
positive	3	Int32
negative	1	Int32
neutral	5	Int32
1	{ 6 attributes }	Object
2	{ 6 attributes }	Object
3	{ 6 attributes }	Object
4	{ 6 attributes }	Object
5	{ 6 attributes }	Object
6	{ 6 attributes }	Object
7	{ 6 attributes }	Object
8	{ 6 attributes }	Object
9	{ 6 attributes }	Object
10	{ 6 attributes }	Object
11	{ 6 attributes }	Object
12	{ 6 attributes }	Object
13	{ 6 attributes }	Object
14	{ 6 attributes }	Object
15	{ 6 attributes }	Object
16	{ 6 attributes }	Object
17	{ 6 attributes }	Object
18	{ 6 attributes }	Object
19	{ 6 attributes }	Object
20	{ 6 attributes }	Object
21	{ 6 attributes }	Object

Figura 4.6: Información comentarios en la base de datos

- **negative** o número de comentarios con valor negativo para este tuit.
- **neutro** o número de comentarios con valor neutro para este tuit.

Gracias a la base de datos, se podría disponer de la información de todos los comentarios desde el primer análisis si se ejecutará cada 7 días.

4.2. Integración tecnológica

En esta sección se va comentar en detalle cada una de las tecnologías implicadas en el sistema, comenzando con una breve introducción para situar al lector y explicando los aspectos más característicos de la implementación y configuración.

4.2.1. Docker

Docker es un proyecto de código abierto que automatiza el despliegue de aplicaciones dentro de contenedores de software, proporcionando una capa adicional de abstracción y automatización de virtualización a nivel de sistema operativo en Linux [Doca]. En este TFG se ha usado para generar un contenedor de Docker para cada uno de los componentes de la arquitectura mencionados anteriormente. Facilita el despliegue, ya que cada una de estos contiene lo necesario para que ese componente funcione. Para que estos contenedores se comuniquen entre ellos se usará Docker-Compose.

4.2.2. Node.js

Node.js es un entorno de ejecución multiplataforma de JavaScript, que utiliza un modelo asíncrono y dirigido por eventos. Esta diseñado para construir aplicaciones escalables [Nod]. Es una tecnología usada principalmente para el desarrollo web e intenta optimizar la entrada y salida de datos, por lo que no se bloquean los procesos.

El servidor se va a desarrollar en esta tecnología por las ventajas que se han comentado anteriormente. Como es fácilmente escalable, en versiones futuras se podría escalar horizontalmente aumentando el número de servidores. Además gracias a la eficiencia de entrada y salida de datos, las comunicaciones con el API de Twitter no se bloquean.

4.2.3. React.js

React.js es una biblioteca de código abierto mantenida por Facebook diseñada para crear interfaces de usuario modulares. Está pensada para ayudar al desarrollo de aplicaciones cuyos datos cambian continuamente. Se basa en componentes permitiendo la separación de la interfaz de usuario en piezas independientes y encapsuladas, pudiendo pensar en cada pieza de forma aislada [Rea].

Esta tecnología se ha usado para el desarrollo de la parte cliente, componetizando la interfaz, que serían cada uno de los elementos relevantes de esta. Esto nos ayuda a reutilizar pequeñas piezas de código haciendo el desarrollo más intuitivo. Al ser una de las tecnologías de frontend más importantes, es sencillo encontrar documentación y soluciones para resolver los problemas encontrados, además de que se integra fácilmente con Node.JS.

4.2.4. MongoDB

MongoDB es una de las bases de datos más importantes dentro de las catalogadas como No SQL o no relacionales[Mon]. Es una base de datos de documentos pensada para optimizar la consistencia, como almacén de gran cantidad de datos. Se ha usado como almacenamiento de toda la información del sistema, ya que se busca velocidad de respuesta ante la recuperación de grandes cantidades de datos.

Se ha usado de base de datos para el almacenamiento de los usuarios del sistema y de la información referente a los datos de Twitter.

4.2.5. Bitbucket

Bitbucket es un servicio de alojamiento web para proyectos que usan el sistema de control de versiones de Git. La gran ventaja que tiene este servicio es su integración con Jira (una herramienta de gestión de proyectos) [Bit].

Se han creado 2 repositorios privados como almacenamiento del código del cliente y del servidor, y el servicio proporciona ayuda al mantenimiento y versionado de estos.

4.2.6. VSCode

Visual Studio Code es un editor de código con soporte multiplataforma y multilenguaje desarrollado por Microsoft. Tiene como características su carácter personalizable, integración con Git y soporte para la depuración. Gracias a un sistema de extensiones y add-ons facilita el desarrollo y ejecución de tareas de Javascript.[Mic]

Se ha utilizado para el desarrollo del código tanto de la aplicación de Node.js como de React.js.

4.3. Interfaz

Se van a comentar las distintas partes que componen la pantalla más importante, la que muestra visualmente las analíticas.

En la Figura 4.7 se pueden observar tres cuadros de información:

- **Datos generales de la cuenta**. El primer cuadro contendría los datos generales de la cuenta relevantes como son los seguidores, seguidos o una pequeña descripción.
- **Rango de fechas**. El segundo cuadro muestra el rango de fechas de los tuits analizados así como el estado del análisis, que puede ser **Done** (cuando la analítica ha finalizado), **Getting Statuses** (si se están analizado los tuits) y **Getting Replies** (si se están analizando los comentarios).
- **Métricas de los tuits**. Sale un conjunto de cuadros con las distintas métricas referentes a los tuits 3.2.

Figura 4.7: Datos generales de cuenta analizada

En la Figura 4.8 se muestra una **nube de tags** referente a los hashtags usados en lo tuits de la cuenta analizada.

Figura 4.8: Ejemplo de nube de tags

La Figura 4.9 muestra la distribución de tuits por día así como el tipo de tuits (con imagen, URL . . .).

La Figura 4.10 hace referencia al análisis de sentimiento de los comentarios sobre los últimos tuits. Se muestra en azul la valoración del tuit, en rojo los comentarios negativos,

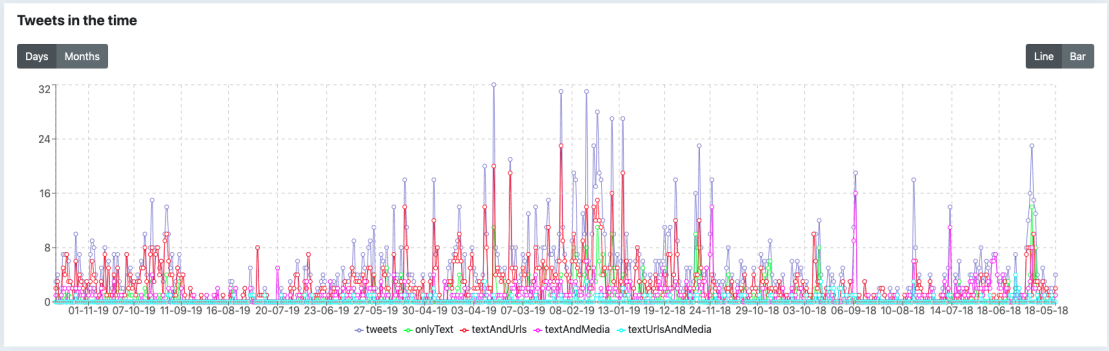

Figura 4.9: Gráfico de uso en el tiempo

en verde los positivos y en gris los neutros. A continuación se muestra de forma numérica las métricas descritas en la Sección 3.2.

Figura 4.10: Análisis de sentimiento sobre las publicaciones

Capítulo 5

Casos de uso

Se van a presentar dos casos de uso para los que este sistema dará un valor añadido a la hora de analizar el uso de las redes sociales por diferentes entidades. En concreto, consideraremos instituciones museísticas y partidos políticos.

5.1. Instituciones museísticas

El primer caso de uso que se plantea es la automatización de una investigación en el campo de la educomunicación web 2.0 planteada por Borja Aso, miembro del Departamento de Didáctica de las Lenguas y de las Ciencias Humanas y Sociales de la Universidad de Zaragoza, y que pretende analizar el uso de redes sociales por parte de instituciones museísticas españolas dedicadas al arte contemporáneo[Aso19].

Hasta ahora, los datos tenían que obtenerse a mano. Con nuestra herramienta, no solo se reducen tiempo y esfuerzo, sino que se aumenta el número de métricas disponibles (por ejemplo, al incorporar análisis de sentimientos).

5.2. Partidos políticos

Como segundo caso de uso, se ha analizado la actividad durante las elecciones generales del 10 de noviembre de 2019 de los seis principales partidos políticos nacionales: Partido Socialista Obrero Español (PSOE), Partido Popular (PP), Vox, Unidad Podemos (UP), Ciudadanos (Cs) y Más País. Comparando las métricas obtenidas, se va a relacionar el uso de las redes sociales para la difusión de su campaña y sus resultados obtenidos (número de escaños).

Se van a hacer valoraciones y conclusiones basadas en los datos que podrían no ser del todo ciertas o solo impresiones, pero estos datos en manos de una analista especialista en el campo le permitirían sacar conclusiones más acertadas o interpretar mejor los datos.

Las Figuras 5.1–5.6 muestran los tuits publicados por día. El primer dato importante es el volumen tan elevado de publicaciones en estos últimos meses: podemos percatarnos de que, habiendo recuperado el máximo de 3200 tuits por cuenta que se puede hacer a la hora de ejecutar la búsqueda, solo hemos recuperado datos de, como mucho, 4 meses. Esta tendencia se cumple para los seis partidos estudiados, pero en el caso de Ciudadanos es de solo un mes. La conclusión que se puede sacar de estos datos es que en estas jornadas electorales ha habido una gran actividad en las redes sociales.

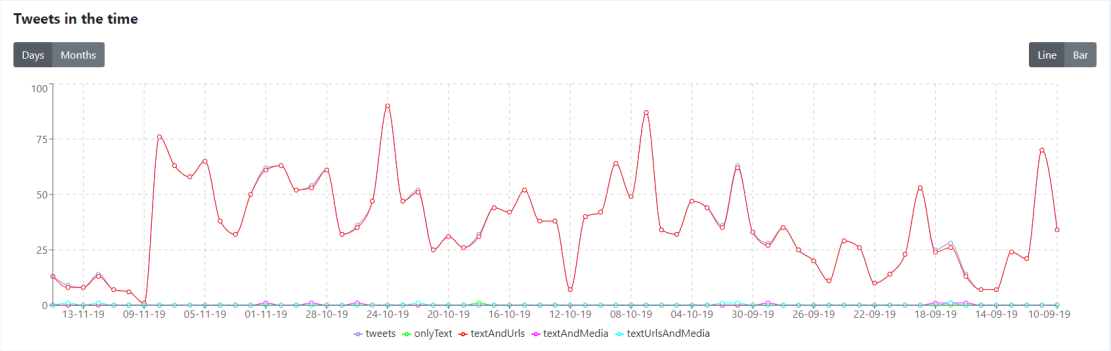

Figura 5.1: Tendencia tuits, PSOE

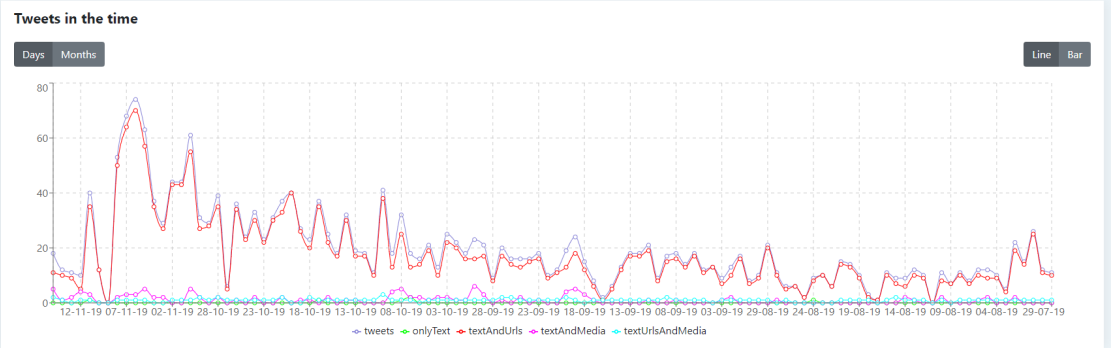

Figura 5.2: Tendencia tuits, PP

Otra tendencia observable es que el día 9 de noviembre (jornada de reflexión) todos bajan sus publicaciones a 0. Los días siguientes, el número de publicaciones de Unidas Podemos, PSOE y Más País es muy bajo o nulo. Sin embargo, Vox, PP y Ciudadanos vuelven a tener presencia en las redes aumentando el número de publicaciones. La conclusión a la que se podría llegar es que el primer grupo de tres son los ganadores o posibles participantes en un futuro gobierno y ya no les resulta tan necesaria una gran presencia en las redes sociales. Por otra parte, el otro grupo de tres que en principio sería la oposición tiene una presencia más marcada en las redes cuestionando las decisiones tomadas por el primer grupo de 3, como ilustra la Figura 5.7, si bien no todos los tuits publicados desde la realización de las elecciones son de este índole.

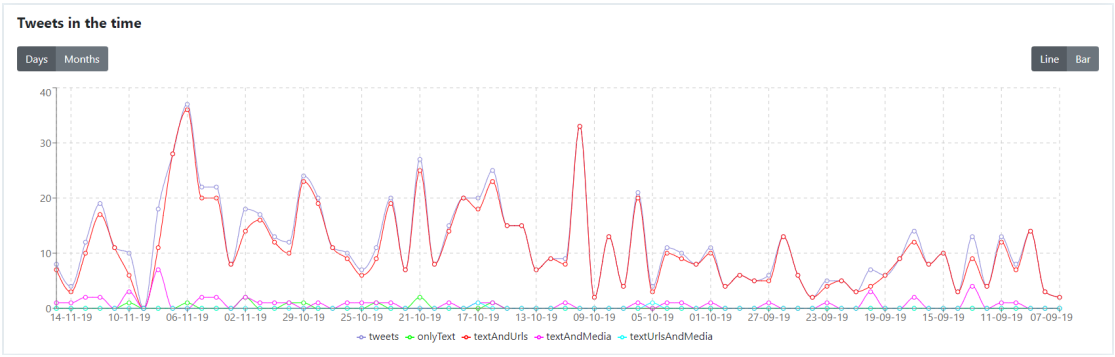

Figura 5.3: Tendencia tuits, Vox

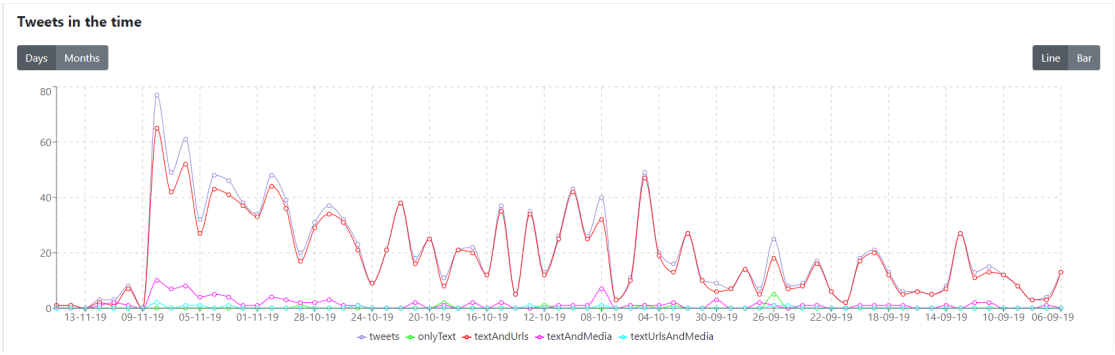

Figura 5.4: Tendencia tuits, Unidas Podemos

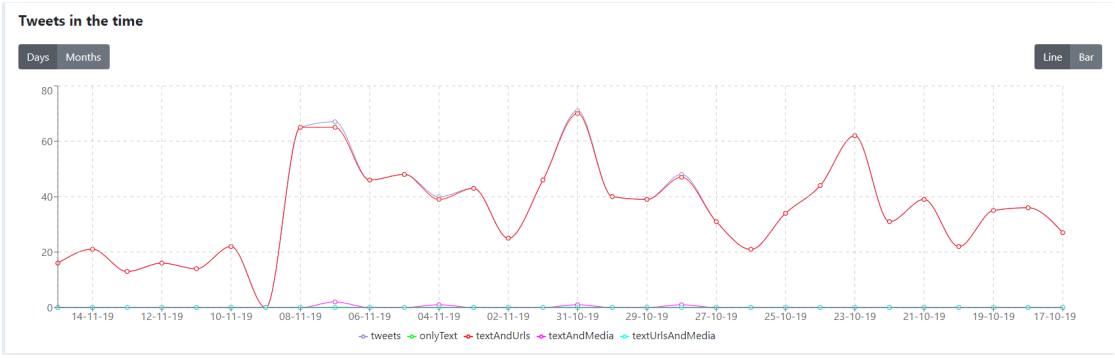

Figura 5.5: Tendencia tuits, Ciudadanos

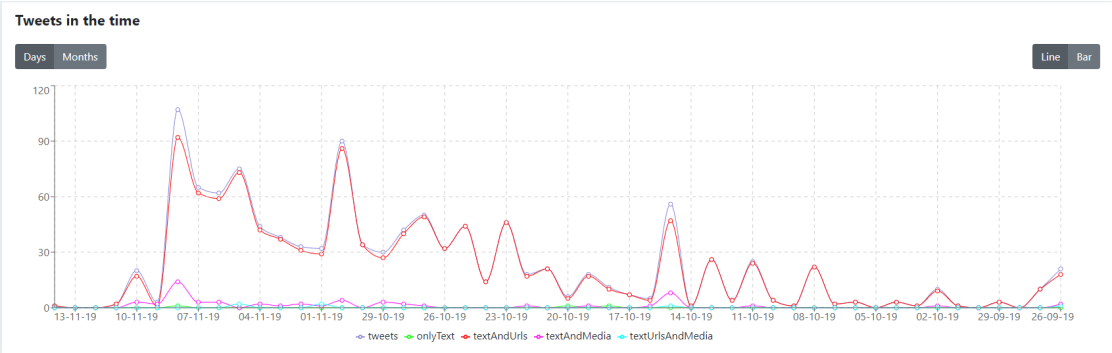

Figura 5.6: Tendencia tuits, Más País

Figura 5.7: Ejemplos de tuits de Vox y PP tras las elecciones

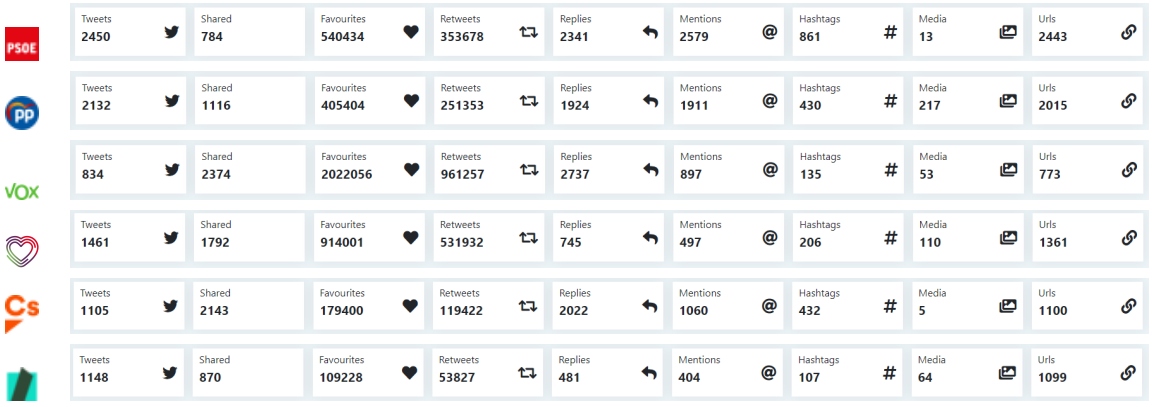

Figura 5.8: Algunas métricas de los seis partidos políticos

Partido	Seguidores	Seguidos	Tuits	Favoritos	Retuits	Escaños
PSOE	721804	13635	103703	540434	353678	120
PP	735769	4511	87533	405404	251353	89
VOX	349342	1134	42609	2022056	961257	52
UP	1434634	1611	105902	914001	531932	35
Cs	520572	92043	144830	179400	119422	10
MP	48050	118	2019	109228	53827	3

Tabla 5.1: Información y métricas de los partidos políticos.

La Figura 5.8 y la Tabla 5.1 muestran algunas métricas de los partidos políticos considerados. A pesar de que cada partido tiene más de 3200 tuits (excepto Más País, con 2019 tuits en el momento de la escritura), se muestra el número de datos que se han podido analizar.

Ciudadanos y Vox apuestan por compartir más tuits de terceros que los suyos propios: Vox 834 publicaciones propias frente a 2374 publicaciones compartidas, y Ciudadanos 1105 frente a 2143. PSOE y PP tienden a publicar sus propios tuits: 2132 frente a 1116 PP y 2450 frente a 784. Unidas Podemos y Más País tienen datos similares de publicaciones propias y compartidas: 1461 frente a 1792, y 1148 frente a 870, respectivamente.

Otro dato muy relevante son la publicaciones con URL: se puede observar que está cerca de igualar al total de tuits en todos ellos, lo que indica que casi todos los tuits que publican contienen URL. Recordemos que para Twitter el concepto de URL incluye vídeos, fotos o enlaces a artículos.

El siguiente dato, y uno de los más relevantes en este análisis, sería el total de interacciones por parte de los usuarios con los tuits de cada cuenta, lo que incluye los favoritos y retuits (ver Tabla 5.1). El partido que más interacción y aceptación ha tenido en sus publicaciones ha sido Vox, ya que de los 6 partidos analizados es el segundo con menos seguidores, pero aún así ha recibido el doble de favoritos y retuits que la segunda con más interacción que es Unidas Podemos, la fuerza política con más seguidores. Al ser el primero y el segundo en datos de interacción se puede argumentar que la gente interactúa más con las fuerzas políticas de los extremos ideológicos.

En la comparación entre ambos partidos de extremos, se percibe una relación entre la subida de escaños por parte de Vox (52) y la bajada de Unidas Podemos (35). La superioridad en favoritos y retuits por parte de Vox, que casi duplica a Unidas Podemos, se ve reflejada en un mayor número de escaños (52 contra 35, que no llega a ser el doble pero sí es una diferencia considerable).

En las dos fuerzas políticas más votadas, PSOE y PP, se pueden observar datos muy similares, siendo PSOE más aceptado. Siendo datos igualados, PSOE tiene más

interacción y aceptación que PP aún teniendo menor número de seguidores. Se puede decir que la lucha entre los dos partidos ha sido igualada pero favorable para el PSOE, lo que también sucedió con los escaños electorales (120 PSOE y 89 PP).

Nos queda analizar dos fuerzas políticas que no han tenido buenos resultados, una que ha bajado en votos y escaños (Ciudadanos) y un partido nuevo que no acaba de arrancar (Más País). Ambas han sido las dos últimas fuerzas en favoritos, retuits y número de escaños (10 para Ciudadanos y 3 para Mas País). Son números especialmente bajos para Ciudadanos, con diez veces más seguidores que Mas País, que siendo una fuerza política nueva ha llegado a tener un volumen de interacción bastante bueno dentro de sus posibilidades.

Gracias a la aplicación se han podido obtener métricas bastante interesantes basadas en un periodo de tiempo, alrededor de unas elecciones generales y previo a la creación de un gobierno. Sería muy complicado observar muchas de estas métricas (y, por lo tanto, comparar estas fuerzas políticas) a simple vista desde Twitter. Además, aunque pase el tiempo y haya otros temas de actualidad al disponer del historial de las búsquedas siempre se podrán recuperar estos resultados y compararlos con el presente.

Aparte de las métricas obtenidas se puede también ver cuales son las menciones y los hashtags más frecuentes. A modo de ejemplo, mostraremos datos de Más País para ilustrar esta funcionalidad. Las Figuras 5.9 y 5.10 muestran nubes de etiquetas. Aunque hay menciones a diferentes cuentas del ámbito político, la mención más frecuente es *ierrejon*, el líder del partido, y el hashtag más usado es *VotaMásPaís*, entre otras palabras relacionadas con el partido.

Figura 5.9: Menciones a Más País

Dada estas nubes de etiquetas se pude pulsar sobre cualquiera y nos mostrará un listado de todos los tuits (entre los que tenemos analizados) que usan la etiqueta pulsada, como se ilustra en la Figura 5.11, donde recuperamos una lista de todos los tuits que usan la mención *ierrejon*. Esto es muy útil, por ejemplo, en casos en que encontremos menciones que no concuerdan aparentemente con el ámbito de una cuenta de Twitter, para poder ver por qué se mencionó. Pulsando sobre la flecha a la derecha se redirigirá al tuit real en Twitter.

En este caso de uso se han mostrado la mayoría de funcionalidades del TFG. Claramente, se podría realizar un estudio similar en otros dominios de aplicación diferentes a los partidos políticos. Por ejemplo, se podría aplicar a empresas alrededor de la fecha del lanzamiento de un producto, a equipos de fútbol en fechas importantes (por ejemplo, un partido de final de Champions League) ...

Figura 5.10: Hashtags usados por Más País

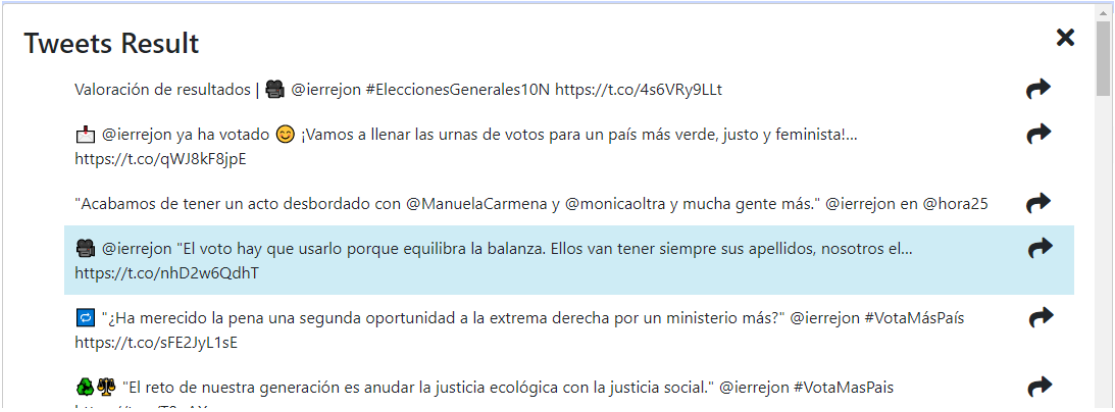

Figura 5.11: Menciones para Más País

Capítulo 6

Conclusiones

En este capítulo se presentan las conclusiones del trabajo, su importancia y algunas líneas de trabajo futuro.

El proceso de análisis de datos es indispensable hoy en día en las corporaciones. También el creciente uso de las redes sociales y el gran volumen de datos que se genera pueden aportar información relevante para crear estrategias de negocio.

En este TFG se ha desarrollado una plataforma que ayude al análisis de las redes sociales, que mejora o complementa las soluciones que ofrecen las principales herramientas similares del mercado. En primer lugar, se ha diseñado un conjunto de métricas de interés. A continuación, se ha desarrollado una herramienta robusta y apta para el marco empresarial de ciertas empresas que quieran estudiar el estado de sus redes sociales y las de la competencia.

La herramienta permite recuperar ciertas métricas, desarrolladas en el marco de este TFG, que a simple vista estas redes sociales no muestran. Algunas de ellas están basadas en el análisis de sentimientos de los comentarios de los usuarios. También se proporciona un historial de datos para medir la evolución de las cuentas analizadas y una funcionalidad para la comparación de cuentas que permite ayudar a ver la diferencia entre marcas o con la competencia. La herramienta solo soporta actualmente Twitter, pero está diseñada de modo que facilite futuras extensiones.

Se han analizado las APIs de diferentes redes sociales, detectando diferentes limitaciones. En particular, ha sido necesario desarrollar algoritmos específicos (por ejemplo, para aumentar el número de comentarios recuperados) y utilizar una base de datos (para tener en cuenta datos de búsquedas anteriores) para superar algunas limitaciones de la API de Twitter.

También se han propuesto un par de cosas de uso, como la gestión de información de museos y el análisis de partidos políticos durante las últimas elecciones generales.

Desde un punto de vista personal, al desarrollar este proyecto no solo he conseguido familiarizarme con el uso de las APIs de las redes sociales sino también con el uso de

nuevas herramientas, metodologías de trabajo y tecnologías del momento, extendiendo los conocimientos adquiridos en asignaturas como Sistemas de información distribuidos, bases de datos, tecnologías web y metodologías ágiles.

Trabajo Futuro La herramienta desarrollada tiene un gran margen de crecimiento y evolución. Por ejemplo, podemos citar los siguientes puntos:

- Integrar más redes sociales de un ámbito similar a Twitter, como Instagram, Facebook y LinkdIn. Esto requiere conseguir cuentas de empresa para poder hacer uso de sus APIs pero el sistema requerirá pequeños cambios ya que está desarrollado para ser lo más genérico posible.
- Evolucionar el sistema para redes sociales de distinto ámbito cómo sería Youtube, la cual requeriría extender nuestro sistema a nuevas funcionalidades.
- Mejorar el sistema de análisis de sentimiento. En esta primera versión se ha utilizado una librería de Javascript basada en el peso de las palabras, y una mejora sería introducir un modelo de Inteligencia Artificial que entendiera el contexto de las frases.
- Mejorar la arquitectura a micro servicios haciendo más fácil la escalabilidad y el desarrollo de las extensiones.
- Crear un servidor independiente para la base de datos, un servidor de datos.

Estos puntos enriquecerían la aplicación proporcionando más funcionalidades y facilitando ser más competitiva en el mercado.

Bibliografía

[Ala17] Dimah Hussain Alahamdi. "Recommender systems based on online social networks - An implicit social trust and sentiment analysis approach". Tesis doct. University of Manchester, UK, 2017.

[Aso19] Borja Aso Morán. "Educomunicación web 2.0 del patrimonio artístico contemporáneo: estudio de caso del museo IAACC Pablo Serrano". Trabajo Fin de Máster. University of Zaragoza, 2019. URL: http://deposita.unizar.es/record/39180.

[Aul] Aula301. *MongoDB lo que deberías de saber antes de utilizarlo - Aula301*. https://aula301.com/cuando-usar-mongodb/. (Accessed on 11/19/2019).

[Bar] Marcelo Barile. *multilang-sentiment - npm*. https://www.npmjs.com/package/multilang-sentiment. (Accessed on 11/17/2019).

[Bit] Bitbucket. *Bitbucket — The Git solution for professional teams*. https://bitbucket.org/. (Accessed on 11/17/2019).

[Caw] Conor Cawley. *What Is Social Media Management? - Tech.co*. https://tech.co/digital-marketing/social-media-management-guide. (Accessed on 11/16/2019).

[Cha] Alvin Chang. *The Facebook and Cambridge Analytica scandal, explained with a simple diagram - Vox*. https://www.vox.com/policy-and-politics/2018/3/23/17151916/facebook-cambridge-analytica-trump-diagram. (Accessed on 11/16/2019).

[Doca] Docker. *Enterprise Container Platform — Docker*. https://www.docker.com/. (Accessed on 11/17/2019).

[Docb] Docker. *Overview of Docker Compose*. https://docs.docker.com/compose/. (Accessed on 11/20/2019).

[Dra18] Mauro Dragoni. "NEUROSENT-PDI at SemEval-2018 Task 3: Understanding Irony in Social Networks Through a Multi-Domain Sentiment Model". En: *Proceedings of The 12th International Workshop on Semantic Evaluation (SemEval@NAACL-HLT 2018)*. 2018, págs. 512-519.

[EN] E&N. *9 beneficios de las redes sociales en los negocios - Revista Estrategia & Negocios*. https://www.estrategiaynegocios.net/marketing/1024491-330/9-beneficios-de-las-redes-sociales-en-los-negocios. (Accessed on 11/16/2019).

[Faca] Facebook. *Business Discovery*. https://developers.facebook.com/docs/instagram-api/guides/business-discovery. (Accessed on 11/17/2019).

[Facb] Facebook. *Facebook for Developers*. https://developers.facebook.com/?no_redirect=1. (Accessed on 11/18/2019).

[GBH09] Alec Go, Richa Bhayani y Lei Huang. *Twitter sentiment classification using distant supervision*. Inf. téc. CS224N Project Report. Stanford University, 2009.

[Gon] Luis Gonçalves. *Qué es la metodología Ágil*. https://luis-goncalves.com/es/que-es-la-metodologia-agil/. (Accessed on 11/16/2019).

[Har] Joshua Hardwick. *Top 100 Most Visited Websites by Search Traffic (as of 2019)*. https://ahrefs.com/blog/most-visited-websites/. (Accessed on 11/15/2019).

[Hoo] Hootsuite. *Social Media Marketing & Management Dashboard - Hootsuite*. https://hootsuite.com. (Accessed on 11/17/2019).

[Lin] LinkedIn. *Getting started with the REST API*. https://developer.linkedin.com/docs/rest-api. (Accessed on 11/17/2019).

[Met] Metricool. *METRICOOL Social media y contenidos en la misma herramienta*. https://metricool.com/es/. (Accessed on 11/17/2019).

[MHI] MHI. *2015-MHI-Sales-Best-Practices-Study*. https://www.slideshare.net/JessicaRando/2015mhisalesbestpracticesstudy-47739840. (Accessed on 11/16/2019).

[Mic] Microsoft. *Visual Studio Code - Code Editing. Redefined*. https://code.visualstudio.com/. (Accessed on 11/16/2019).

[Mon] Mongo DB. *La base de datos líder del mercado para aplicaciones modernas — MongoDB*. https://www.mongodb.com/es. (Accessed on 11/16/2019).

[Nod] Node.js. *About Node.js*. https://nodejs.org/en/about. (Accessed on 11/16/2019).

[Rea] Reactjs. *React – Una biblioteca de JavaScript para construir interfaces de usuario*. https://es.reactjs.org/. (Accessed on 11/16/2019).

[Rev] Revive Digital. *Most Popular Social Media Networks (Updated for 2019) — Revive Digital*. https://revive.digital/blog/most-popular-social-media. (Accessed on 11/16/2019).

[Rib] Ester Ribas. *Qué es Api Rest y por qué debes de integrarla en tu negocio ¡Descúbrelo!* https://www.iebschool.com/blog/que-es-api-rest-integrar-negocio-business-tech/. (Accessed on 11/16/2019).

[Roc] RockContent. *Historia de las redes sociales: conoce de dónde surgieron y su evolución*. (Accessed on 11/14/2019). URL: https://rockcontent.com/es/blog/historia-de-las-redes-sociales.

[Rou] Margaret Rouse. *¿Qué es Interfaz de programación de aplicaciones (API) ?* https://searchdatacenter.techtarget.com/es/definicion/Interfaz-de-programacion-de-aplicaciones-API. (Accessed on 11/20/2019).

[Sam] Sampol. *Qué es el stack MERN de JavaScript*. https://platzi.com/blog/que-es-mern-stack-javascript/. (Accessed on 11/16/2019).

[Sea] Twitter Search. *Standard search — Twitter Developers*. https://developer.twitter.com/en/docs/tweets/search/overview/standard. (Accessed on 11/17/2019).

[Sim] Carlos Vieira Simões. *dissertacao.pdf*. https://fenix.tecnico.ulisboa.pt/downloadFile/844820067125238/dissertacao.pdf. (Accessed on 11/17/2019).

[Sob] José Carlos Sobrino. *Análisis de sentimientos en Twitter*. http://openaccess.uoc.edu/webapps/o2/bitstream/10609/81435/6/jsobrinosTFM0618memoria.pdf. (Accessed on 11/17/2019).

[Stu] Social Studio. *Social Studio*. https://help.salesforce.com/articleView?id=mc_ss_social_studio.htm&type=5. (Accessed on 11/18/2019).

[Tra] Traslaniebla. *Organizate con Kanban — Tras la niebla*. URL: http://www.traslaniebla.com/2014/03/20/organizate-con-kanban/ (visitado 26-06-2018).

[Twia] Twitter. *Access tokens*. https://developer.twitter.com/en/docs/basics/authentication/guides/access-tokens. (Accessed on 11/17/2019).

[Twib] Twitter. *Follow, search, and get users*. https://developer.twitter.com/en/docs/accounts-and-users/follow-search-get-users/api-reference/get-users-search. (Accessed on 11/17/2019).

[Twic] Twitter. *GET statuses/user_timeline — Twitter Developers*. https://developer.twitter.com/en/docs/tweets/timelines/api-reference/get-statuses-user_timeline. (Accessed on 11/17/2019).

[Twid] Twitter. *Twitter Analytics*. https://analytics.twitter.com/about. (Accessed on 11/17/2019).

[Wika] Wikipedia. *Classmates.com*. (Accessed on 11/14/2019). URL: https://en.wikipedia.org/wiki/Classmates.com%5C#Users_and_ranking_among_other_social_networking_sites.

[Wikb] Wikipedia. *Front-end y back-end*. https://es.wikipedia.org/wiki/Front-end_y_back-end. (Accessed on 11/20/2019).

[Wikc] Wikipedia. *Kit de desarrollo de software*. https://es.wikipedia.org/wiki/Kit_de_desarrollo_de_software. (Accessed on 11/20/2019).

[Wikd] Wikipedia. *MongoDB*. https://es.wikipedia.org/wiki/MongoDB. (Accessed on 11/20/2019).

[Wike] Wikipedia. *Node.js*. https://es.wikipedia.org/wiki/Node.js. (Accessed on 11/20/2019).

[Wikf] Wikipedia. *OAuth*. https://es.wikipedia.org/wiki/OAuth. (Accessed on 11/17/2019).

[Wikg] Wikipedia. *SixDegrees.com*. https://en.wikipedia.org/wiki/SixDegrees.com. (Accessed on 11/15/2019).

[Wikh] Wikipedia. *Social network*. (Accessed on 11/14/2019). URL: https://en.wikipedia.org/wiki/Social%5C_network.

[Wiki] Wikipedia. *Transferencia de Estado Representacional*. https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional. (Accessed on 11/20/2019).

[Wikj] Wikipedia. *World Wide Web*. https://en.wikipedia.org/wiki/World_Wide_Web. (Accessed on 11/16/2019).

[Wikk] Wikipedia. *YAML*. https://es.wikipedia.org/wiki/YAML. (Accessed on 11/20/2019).

Apéndice A

Horas invertidas

El desarrollo de trabajo se ha prolongado durante 4 meses, se ha controlado en este periodo el tiempo empleado para cada una de las tareas. En la figura A.1 se muestra un diagrama de Gantt que representa el esfuerzo dedicado desglosado por meses y tareas.

El trabajo se compone de las siguientes tareas:

1. Evaluación de las herramientas de análisis sobre redes sociales y de lo que ofrecen las diferentes APIs de las redes sociales que se han seleccionado.
2. Análisis y diseño inicial, donde se analizan los requisitos del sistema y las particularidades que nos encontramos y se proponen las métricas.
3. Implementación y pruebas, que consta de cuatro sprints en las que se han desarrollado, probado y revisado las tareas asignadas a cada uno.
4. Documentación de la realización del proyecto.

Este diagrama es una aproximación al desarrollo y una interpretación de los hitos. En total, se han invertido 300 horas, en torno a un 20 % han sido invertidas a la generación de la documentación y el resto a las fases de implementación, análisis y evaluación.

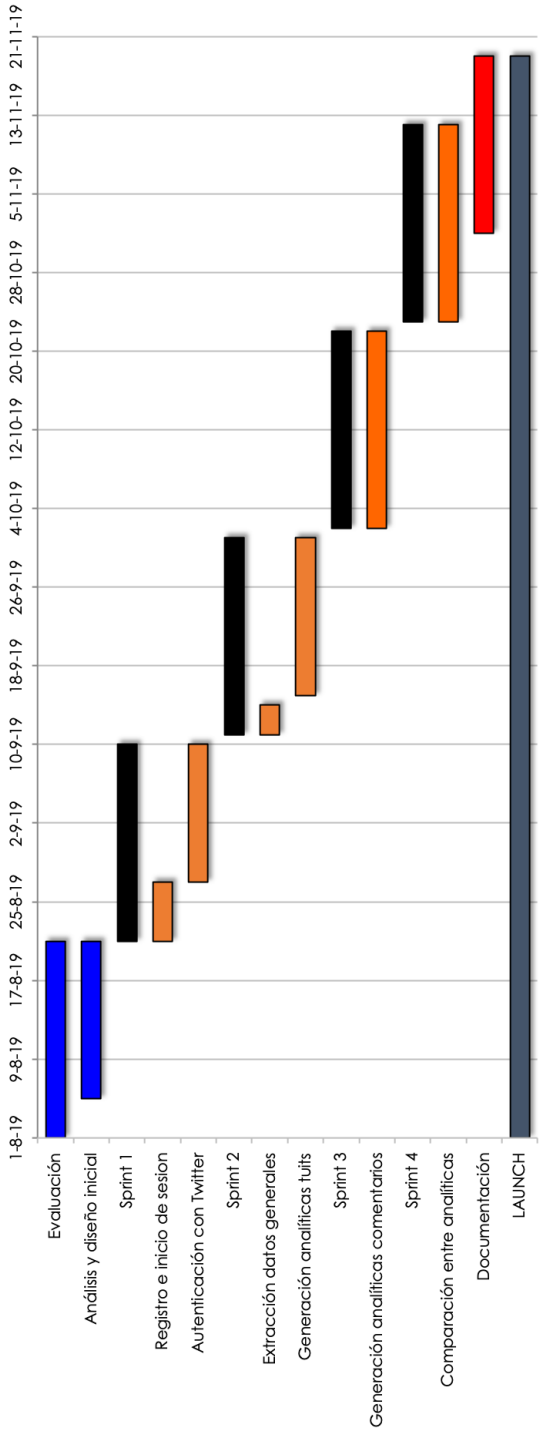

Figura A.1: Diagrama de Gantt con el esfuerzo invertido en días

Apéndice B

Análisis empresarial

El mercado actual se encuentra repleto de diversas redes sociales, orientadas a diversos públicos, por lo que no debe sorprender el hecho de que exista un ecosistema de empresas y herramientas ofreciendo servicios para usuario y clientes. Este hecho sugiere preguntas como: ¿Qué marca la diferencia entre los servicios? ¿Qué producto o solución de los evaluados se encuentra en el liderazgo? ¿Cuáles son los mejores proveedores para mi proyecto?.

Debido a que se va a centrar en el uso de la red social Twitter, a continuación se presenta una comparativa de una selección relevante de herramientas segmentadas según su foco y funcionalidad. Hay que destacar que no se ha podido probar ninguna funcionalidad actual de pago, y ciertos elementos se basan en lo que se ha obtenido de su documentación.

Twitter Analytics. Es la herramienta nativa de la propia plataforma Twitter (Figura B.1). Está centrada en medir la interacción y el éxito de los tuits[Twid]. Es de obligada mención, pero no aporta ninguna métrica relevante para un análisis más profundo, y mucho menos extraer un sentimiento, pues está orientada al usuario con menos necesidades de negocio.

Hootsuite. Es una herramienta centrada principalmente en gestionar todas tus redes sociales desde una sola plataforma permitiendo funcionalidades como programación y respuesta automática de mensajes. Aunque se presenta como un producto orientado a gestión y no a análisis, la herramienta presenta una generación de métricas con una pequeña parte de análisis de sentimiento como se puede observar en la Figura B.2. Además de ser una aplicación de pago, el escaso análisis presentado dista bastante del objetivo de este proyecto.

Metricool. A diferencia de una herramienta de gestión como Hootsuite, Metricool está más orientada a presentar métricas tanto en tiempo real como en un rango temporal a diferencia de las dos herramientas anteriores (ver Figura B.3). Por contra, no provee de análisis de sentimiento y está solo centrada en métricas básicas como el número de impresiones o respuestas.

Salesforce Social Studio. Es una solución integral que permite gestionar, programar, crear y monitorizar publicaciones (Figura B.4). Puede organizar las publicaciones por marca, región o múltiples equipos y personas en una interfaz unificada [Stu]. Además posee la característica de poder realizar análisis de sentimiento para campañas publicitarias, de la marca, de los productos de la marca pero no el análisis individual de las publicaciones. También está centrada en gestionar y obtener métricas de las cuentas de tus redes sociales y no de las de otros usuarios. Social Studio forma parte del ecosistema de Salesforce y el precio oscila entre 100 y 600 euros al mes por cuenta.

Como se ha visto, la mayoría de las herramientas del mercado se encuentran orientadas a gestión y sólo Social Studio presenta un análisis de sentimiento como se requiere en este TFG. Aún así, dicha herramienta tampoco provee análisis individuales de publicaciones así como de las publicaciones de otros usuarios/competidores. El presupuesto que una herramienta de esa envergadura requeriría y las limitaciones en ciertas funcionalidades justifican el desarrollo de una aplicación propia.

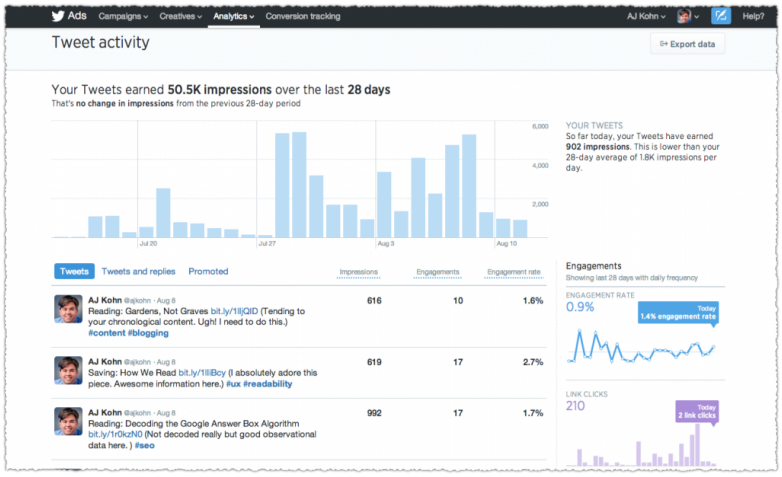

Figura B.1: Twitter Analytics [Twid]

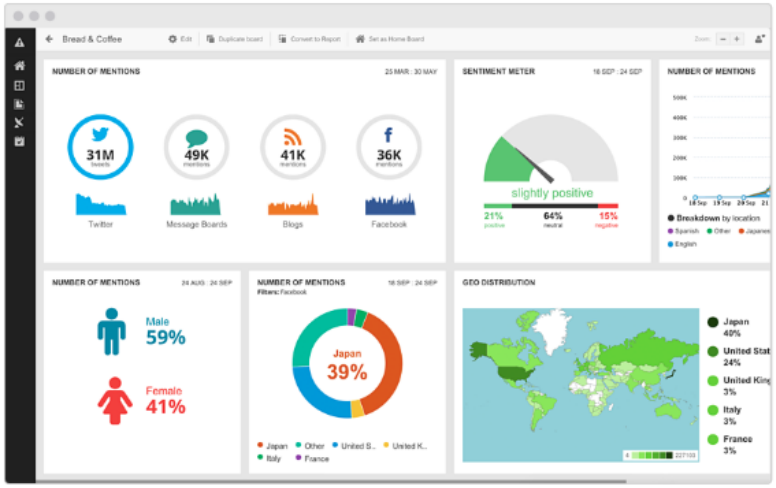

Figura B.2: Hootsuite [Hoo]

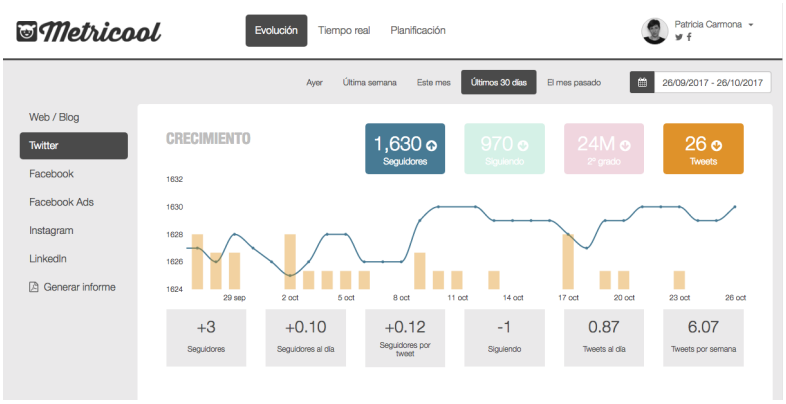

Figura B.3: Metricool [Met]

Figura B.4: Salesforce Social Studio [Hoo]

Apéndice C

Evolución del sistema y Sprints

C.1. Evolución del sistema

En este TFG nos hemos centrado en optimizar el modelo de datos. Con pocos datos un modelo de datos no óptimo puede funcionar correctamente, pero cuando el volumen es alto empieza a haber demoras y tiempos de carga muy largos.

El modelo de datos ha ido evolucionando al mismo ritmo que ha evolucionado el algoritmo de obtención de tuits ya que ambos son dependientes.

C.1.1. Sprint 1

La funcionalidad del primer sprint era muy básica: solo incluía un algoritmo para recuperar la información de la cuenta de usuario de Twitter, una lista de tuits y guardarlo en la base de datos. El servidor solo tenía una colección que no se encargaba de generar las métricas de los tuits, simplemente se encargaba de la obtención y almacenamiento de los datos. El cliente era el encargado de pedir los datos comentados y generar las métricas, encargándose de gran parte de la lógica de negocio, por lo que las transiciones entre pantallas se ralentizaban llegando a costar entre 300 y 400 milisegundos. El sistema tenía los siguientes problemas de rendimiento:

- Tiempo excesivo desde que se ordenaba analizar una cuenta desde el cliente hasta que se mostraba la información. Dependiendo del número de tuits de la cuenta, había una media de 5 segundos para cuentas con números bajos de tuits y 14 segundos en cuentas con un número total de tuits próximo a 3200. En ese momento aún no estaba implementada la recuperación de comentarios sobre los tuits, que habría aumentado el tiempo aun más.
- Se guardaban de cada análisis todos los tuits recuperados junto la información general de la cuenta analizada, así que si se ejecutaba varias veces una analítica

sobre una misma cuenta, podían almacenarse tuits repetidos. Esto generaba mucho volumen de datos, ocupando mucho espacio en la base de datos y ralentizando las consultas.

- Tiempo excesivo de carga para recuperar el historial de analíticas. El principal problema es que para mostrar un historial de analíticas solo se necesita mostrar nombre de la cuenta, descripción y fecha del análisis, pero cada uno de los registros de analíticas tenía la colección de tuits guardada, aumentando el tiempo de respuesta.

La conclusión de esta primera aproximación era que el sistema era poco escalable: con pocas analíticas realizadas el sistema funcionaba fluido pero conforme se iban creando nuevas analíticas el tiempo de respuesta y el espacio ocupado en la base de datos se incrementaban linealmente.

C.1.2. Sprint 2

El primer problema que se quería resolver era reducir el tiempo de respuesta para recuperar datos de las analíticas. Para ello, se creó otra colección llamada Tuit para separar los datos generales de una cuenta de sus tuits. Esto tiene varias ventajas:

- Al tener dos colecciones separadas se podía guardar la información de la cuenta de un usuario y retornar el flujo del sistema al cliente en cuestión de 1 segundo, ya que para obtener esta información solo es necesaria una petición a la API de Twitter. Además, se ejecutaba de forma asíncrona el algoritmo de obtención de tuits y el almacenamiento en la base de datos.
- Poder recuperar el historial de datos inmediatamente en cuestión de milisegundos (se han cronometrado 200 ms) al disponer de las colecciones separadas: al recuperar los datos solo se usa la colección Analíticas (en el primer sprint se recuperaban todas las colecciones de tuits de todas las analíticas).

En este sprint se ha incrementado el tiempo de análisis de los tuits ya que se ha introducido la obtención de los comentarios y su análisis de sentimiento. Esta nueva funcionalidad añadió los siguientes problemas:

- El proceso entero de recuperación de tuits y comentarios podía llegar a costar varios minutos dependiendo del número de tuits que haya en el rango de los 7 días anteriores.
- Como se empezó guardando toda la lista de tuits con los comentarios asociados, a veces Mongo no podía guardar el objeto entero directamente y era necesario dividirlo en partes aumentando la complejidad de desarrollo y mantenimiento del código.
- El cliente seguía calculando las analíticas.

C.1.3. Sprint 3

En este sprint se intenta mover la lógica de cálculo de las analíticas al servidor. Además, se guardan solo los datos relevantes de los tuits, y de los comentarios solo los datos del análisis de sentimiento y el id del tuit al que están asociados. Así, se disminuye el tamaño de los registros y Mongo no tiene problemas para guardar estos datos. Para esto se crea otra colección en la base de datos que almacena las métricas generadas por los tuits. Esta es la lista de ventajas:

- Al mover el análisis de los tuits al servidor y guardarlos directamente en la base de datos, solo era necesario calcularlos una vez y no cada vez como que se requería de ellos como en el cliente. El tiempo de generarlos a la vez que se recupera la lista de tuits en las analíticas no se ve incrementado de forma notoria y en el cliente ahora pasa de 400 ms de latencia cada vez que se recalculaba a ser instantáneo.
- En el cliente al no tener que recuperar datos con todos los tuits la petición pasa a ser bytes de datos y no megabytes.
- Al no guardar casi información de los comentarios asociados a los tuits, solo métricas numéricas del análisis de sentimiento, el tamaño ocupado en la base de datos por cada registro es notablemente inferior.

En ese momento por cada analítica realizada se guardaban 3 colecciones, la primera almacenava la información general de la cuenta, la segunda los tuits asociados y la tercera las métricas.

C.1.4. Sprint 4

En el último sprint se ha optimizado el código para que cuando se hayan realizado ya búsquedas anteriores de una cuenta de usuario no se vuelva a guardar una lista entera con todos sus tuits, sino que solo añadirá los nuevos tuits recuperados a la base de datos y, si no hay nuevos, se mantendrán los anteriores.

Esto es beneficioso porque se elimina la redundancia de miles de datos, liberándose mucha carga y espacio en la base de datos, al tener solo un registro de todos los tuits por cuenta de usuario. También se evita hacer concatenaciones a la hora de hacer consultas, disminuyendo la complejidad del código.

Apéndice D

Pruebas

En este capítulo se va a hablar de las pruebas que se han realizado al sistema, tanto pruebas unitarias como de aceptación.

D.1. Pruebas unitarias

En esta sección se va a describir el plan de pruebas unitarias. Se quiere probar que cada pieza de código tiene el comportamiento esperado, que dada una entrada de datos se devuelva una salida de datos esperada. El plan de pruebas unitarias ha sido ejecutado en el cliente de React.js.

D.1.1. Cliente

Se han realizado las pruebas sobre todos los componentes y también sobre los servicios. Los componentes están compuestos por código Javascript y HTML, dados unos parámetros de entrada se debe generar un HTML de salida.

Para realizar las pruebas se ha utilizado dos bibliotecas de Javascript: Jest y Enzyme. Jest[1] es una herramienta de pruebas desarrollada por Facebook enfocada a la simplicidad, encargada de realizar pruebas sobre componentes web y que facilita funciones para ello. Enzyme[2] es una librería que ayuda al desarrollo y añade funcionalidades a Jest.

La prueba contiene un total de 53 pruebas unitarias que han concluido satisfactoriamente, repartidas en 10 colecciones de pruebas que corresponden a cada uno de los componentes probados (Ver Figura D.1).

La Figura D.1 muestra una tabla de resultados para todos los componentes de la aplicación, dividida en 5 columnas:

[1] `http://jestjs.io`
[2] `http://github.com/airbnb/enzyme`

```
Test Suites: 10 passed, 10 total
Tests:       53 passed, 53 total
Snapshots:   0 total
Time:        4.083s
Ran all test suites.
```

Figura D.1: Total de pruebas unitarias

1. **% Stms.** Porcentaje de instrucciones por las que se ha ejecutado la prueba.
2. **% Branch.** Porcentaje de estructuras de control por las que la prueba se ha ejecutado completamente. Por un ejemplo, en un if-else, que se hayan probado todas las instrucciones del if y todas las del else.
3. **% Func.** Porcentaje de funciones donde se ha ejecutado la prueba en todas sus instrucciones.
4. **% Lines.** Porcentaje de líneas donde se ha ejecutado la prueba.
5. **Uncovered Lines.** Número de líneas donde la prueba no se ha ejecutado.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Lines
All files	91.8	74.63	86.49	95.65	
src/services	100	100	100	100	
twitter.tsx	100	100	100	100	
src/components	91.03	73.23	85.92	95.22	
app.tsx	89.74	58.82	84.62	94.12	4,5
auth.jsx	91.67	78.95	83.33	95.65	10,25
instructions.tsx	100	100	100	100	
card.jsx	100	100	100	100	
compare.jsx	100	100	100	100	
grid.jsx	90.48	72	84.62	95	4,5
history.jsx	100	100	100	100	
request.jsx	87.1	64.71	81.82	93.1	4,5
tweet-metrics.jsx	84.09	66.67	68.75	87.8	4,5,30,39,44

Figura D.2: Tabla de cobertura de código en el cliente

Como se puede observar, el porcentaje de las cuatro primeras columnas es bastante alto, baja un poco la cobertura en la prueba de estructuras de control y de funciones pero aún así se mantiene por encima del 50 %, por lo que se ha probado casi todo el código de la aplicación.

D.1.2. Servidor

El servidor no tiene un plan de pruebas unitarias ya que sus funciones interactúan con la con la API de Twitter, que es un sistema externo, y la base de datos. Para esta parte se ha decidido realizar otro tipo de pruebas, descritas en la Sección D.2.

D.2. Pruebas de aceptación - UAT

Las pruebas de aceptación forman parte de las últimas etapas antes de la liberación de versiones del sistema. Estas pruebas las debería hacer el usuario final de la aplicación pero en este caso las hará manualmente el desarrollador. Gracias a estas pruebas se podrá probar no solo el servidor (que no había pasado un plan de pruebas unitarias), sino también el cliente, ya que las pruebas se hacen desde la perspectiva de un usuario utilizando la aplicación **Pruebasd33**.

En las siguientes subsecciones se van a detallar las pruebas a realizar. Cada una de ellas se compondrá de una descripción, unas precondiciones, unos pasos a seguir y el estado de la prueba.

D.2.1. UAT - registro e inicio de sesión

Se ha probado que un usuario se ha podido registrar y posteriormente iniciar sesión en el sistema.

Lista de precondiciones:

- No hay ninguna sesión iniciada de ningún usuario.
- El usuario con el que se va a hacer la prueba no está registrado en el sistema.
- Iniciar el flujo desde la pagina principal que es la de inicio de sesión.

Pasos a seguir:

1. Pulsar sobre el hipervínculo **Sign up**.
2. Rellenar los campos **Username**, **Email**, **Password** y **Repeat Password**; los dos últimos valores deben coincidir.
3. Pulsar sobre el botón **Signup**.
4. En este momento ya se habrá iniciado sesión: por defecto, cuando te registras en la aplicación satisfactoriamente se inicia sesión.
5. Pulsar sobre el botón del la esquina izquierda **Logout**.

6. Nos encontraremos en la pantalla de inicio de sesión: rellenar el campo **Email** y el campo **Password** con los datos que se han usado en el registro.
7. Pulsar el botón **Login**.
8. Se debería estar en la pantalla principal de la parte interna de la aplicación **Home**.

La prueba se ha realizado con resultado satisfactorio a fecha 16/11/2019.

D.2.2. UAT - Iniciar sesión en la aplicación de Twitter

Se ha probado que un usuario registrado en el sistema se ha podido registrar en la aplicación de Twitter.

Lista de precondiciones:

- Disponer de un usuario registrado en nuestro sistema.
- Haber iniciado sesión con ese usuario.
- Estar en la pantalla **Home**.

Pasos a seguir:

1. Pulsar sobre el hipervínculo del cuadro de navegación izquierdo llamado **Configuration**.
2. Pulsar sobre el botón Login with Twitter.
3. La acción anterior habrá redirigido una pantalla de inicio de sesión de Twitter.
4. Introducir los datos que solicita de una cuenta válida y pulsar al botón de validar.
5. Se habrá vuelto a redirigir a la pantalla **Configuration** de nuestro sistema.
6. En la tabla de cuentas debería salir una cuenta de Twitter, lo que indica que se han recuperado los datos de acceso correctamente.

La prueba se ha realizado con resultado satisfactorio a fecha 16/11/2019.

D.2.3. UAT - Ejecutar una analítica con sugerencia de usuario

Se ha probado el flujo completo de generación de analíticas facilitando una cadena de texto y recuperando un conjunto de datos que concuerde con la cadena de texto facilitado.

Lista de precondiciones:

- Disponer de un usuario registrado en nuestro sistema.
- Haber iniciado sesión con ese usuario.
- Estar en la pantalla **Home**.
- Disponer de un inicio de sesión en la aplicación de Twitter.

Pasos a seguir:

1. En el cuadro de texto escribir una cadena de texto (Ej: Real Zaragoza) y pulsar la tecla **Intro**.
2. Saldrá un mosaico de datos con diversas cuentas de Twitter, seleccionar la primera.
3. Se redirigirá a la pantalla de **Resultados** y nos aparecerá un cuadro con información de la cuenta (seguidores, seguidos y tuits). El segundo cuadro de información incluirá un texto **Status of tweets:** [**state**]. Si **state** es distinto a **Done**, aún no tiene los datos de las métricas calculado: pulsar sobre el botón a la derecha Refrescar Información.
4. Si **state** es distinto a Done repetir el paso 3, si es igual a Done las analíticas se habrán realizado correctamente. Se podrán observar datos numéricos como tuits, retuits, gráficas...

La prueba se ha realizado con resultado satisfactorio a fecha 16/11/2019.

D.2.4. UAT - Ejecutar una analítica sin sugerencia de usuario

Se ha probado el flujo completo de generación de analíticas usando @ + nombre cuenta de usuario en el campo de texto para ejecutar la analítica directamente sin necesidad de devolver un conjunto de datos recomendados que concuerden con la cadena de texto facilitada.

Lista de precondiciones:

- Disponer de un usuario registrado en nuestro sistema.
- Haber iniciado sesión con ese usuario.
- Estar en la pantalla **Home**.
- Disponer de un inicio de sesión en la aplicación de Twitter.

Pasos a seguir:

1. En el cuadro de texto escribir una cadena texto que empiece por @[nombre de usuario] nombre usuario es el identificador de una cuenta de twitter (por ejemplo, **@RealZaragoza**) y pulsar **Intro**.
2. Se redirigirá a la pantalla de **Resultados** y nos aparecerá un cuadro con información de la cuenta. El segundo cuadro de información incluirá un texto **Status of tweets:** [**state**]; si **state** es distinto a Done, pulsar sobre el botón "Refrescar Información".
3. Si **state** es distinto a Done repetir el paso 2, si es igual a Done las analíticas se habrán realizado correctamente.

La prueba se ha realizado con resultado satisfactorio a fecha 16/11/2019.

D.2.5. UAT - Acceder a una analítica del historial

Se va a proceder acceder a la información alguna analítica que dispongamos en el historial.

Lista de precondiciones:

- Disponer de un usuario registrado en nuestro sistema.
- Haber iniciado sesión con ese usuario.
- Estar en la pantalla **Home**.
- Que el sistema disponga de analíticas realizadas anteriormente para poder disponer de un historial.

Pasos a seguir:

1. Pulsar sobre el hipervínculo del cuadro de navegación izquierdo llamado **Historial**.
2. Seleccionar un registro de la lista.
3. Se redirigirá a la pantalla de **Resultados** y nos aparecerá un cuadro con información de la cuenta (seguidores, seguidos, tuits) e información como retuits, gráficas...

La prueba se ha realizado con resultado satisfactorio a fecha 16/11/2019.

D.2.6. UAT - Comparación de una analítica con una nueva analítica

Se ha probado el flujo de comparar una analítica con otra que se genere en el momento sobre cualquier cuenta de Twitter.

Lista de precondiciones:

- Disponer de un usuario registrado en nuestro sistema.
- Haber iniciado sesión con ese usuario.
- Estar en la pantalla de **Resultados** con una analítica cargada.

Pasos a seguir:

1. Pulsar sobre el botón Compare y la pantalla se dividirá en 2.
2. Seleccionar la opción Nueva Analítica.
3. En el cuadro de texto escribir una cadena texto que empiece por **@[nombre de usuario]** y pulsar **Intro**. Como alternativa, se prueba escribir una cadena de texto cualquiera, pulsar **Intro** y seleccionar la primera cuenta del mosaico de datos mostrado.
4. Se cargará en la parte derecha una analítica como la de la parte izquierda pero con los datos de la nueva analítica y unos datos intermedios que son las diferencias de datos numéricos remarcadas. Por ejemplo, si la primera cuenta tiene 10 tuits y la segunda 12, la diferencia sería -2 ya que la cuenta dominante es la primera.

La prueba se ha realizado con resultado satisfactorio a fecha 16/11/2019.

D.2.7. UAT - Comparación de una analítica con una analítica del historial

Se ha probado el flujo de comparar una analítica con otra que se genere en el momento sobre cualquier cuenta de Twitter.

Lista de precondiciones:

- Disponer de un usuario registrado en nuestro sistema.
- Haber iniciado sesión con ese usuario.
- Estar en la pantalla de **Resultados** con una analítica cargada.
- Que el sistema disponga de analíticas realizadas anteriormente para poder disponer de un historial.

Pasos a seguir:

1. Pulsar sobre el botón Compare y la pantalla se dividirá en 2.
2. Seleccionar la opción **Historial** de Analíticas.
3. Seleccionar una de las analíticas del historial.

4. Se cargará en la parte derecha una analítica como la de la parte izquierda pero con los datos de analítica del historial y unos datos intermedios que son las diferencias de datos numéricos remarcadas.

La prueba se ha realizado con resultado satisfactorio a fecha 16/11/2019.

Apéndice E

Despliegue

E.1. Despliegue local

En esta sección se va a presentar cómo instalar el sistema en una máquina local ya sea par su uso o para continuar su desarrollo.

Los requisitos del sistema son:

1. Tener instalado MongoDB.
2. Tener instalado Node.
3. Descargar el código del proyecto cliente y del proyecto servidor de Bitbucket.

Una vez tenemos todo esto preparado, deberemos descargar las librerías necesarias tanto para el cliente como para el servidor.

Para instalar tanto el servidor como el cliente, seguimos los siguientes pasos:

1. Iniciamos la terminal del sistema en el que estemos, y nos desplazaremos hasta la carpeta del proyecto del servidor/cliente que hemos descargado anteriormente.
2. Instalamos las dependencias necesarias para el sistema, escribiendo en la consola lo siguiente:

   ```
   $ npm install
   ```

   Este comando instalará las librerías necesarias para que el servidor/cliente funcione, el comando npm está incluido en la instalación de Node.
3. Lo siguiente es ejecutar el sistema:

```
$ npm start
```

Con esto el backend (al instalar el servidor) y el frontend (al instalar el cliente) ya estarán funcionando. Antes de seguir, se debe comprobar que la base de datos Mongo está activa y se tiene acceso.

Para acceder a la aplicación, se debe acceder a la dirección web http://localhost:8080

E.2. Despliegue en una máquina

Para esta sección se ha automatizado el despliegue de la aplicación usando Docker. Evidentemente, se deberá tener en la máquina instalado Docker, pero no será necesario tener instalado ni MongoDB, ni NodeJS, ya que Docker provee imágenes con esas arquitecturas.

La Figura E.1 muestra un archivo de definición de Docker Compose para instalar todo el sistema. Levantará 3 imágenes independientes de Docker, cada una de ellas con uno de los componentes del sistema. Esto nos facilitará que con un comando se pueda tener todo el sistema en funcionamiento.

En la imagen se puede observar que se especifican 3 servicios:

1. Mongo va a iniciar una imagen reducida de Linux en un contenedor de Docker con MongoDB instalado y enlaza el puerto que usa Mongo por defecto con el del contenedor que se ha generado, para que pueda ser accesible desde fuera de este.
2. Node va a iniciar la imagen en un contenedor pero en este caso con Node instalado. En este es necesario que en la máquina que va a ejecutar este script se tenga el proyecto del servidor copiado en la carpeta que se especifica en 'working_dir'; en el ejemplo deberíamos tener el código en la carpeta '/home/rgabas/backend'. Iniciará el sistema del servidor e igual que Mongo lo expondrá pero en este caso en el puerto 8081.
3. React es igual que el servicio de Node, ya que para ejecutar React también se necesita una imagen con Node instalado y tener copiado el proyecto cliente en la carpeta '/home/rgabas/frontend'. Se inicia el sistema del cliente en el puerto 8080.

Para poder arrancar esto se usará el comando:

```
docker-compose [ruta del fichero de configuración]
```

Una vez ejecutado este comando ya se habrá iniciado el sistema.

```yaml
version: '3.1'

services:
  mongo:
    image: mongo
    restart: always
    ports:
      - 27017:27017
    environment:
      ME_CONFIG_MONGODB_ADMINUSERNAME: rgabas
      ME_CONFIG_MONGODB_ADMINPASSWORD: passwd
  node:
    image: "node:11"
    user: "rgabas"
    working_dir: /home/rgabas/backend
    environment:
      - NODE_ENV=production
    volumes:
      - ./:/home/rgabas/backend
    expose:
      - "8081"
    command: "npm start"
  react:
    image: "node:11"
    user: "node"
    working_dir: /home/rgabas/frontend
    environment:
      - NODE_ENV=production
    volumes:
      - ./:/home/rgabas/app
    expose:
      - "8082"
    command: "npm start"
```

Figura E.1: Configuración Docker Compose para el sistema desarrollado

Apéndice F

Glosario

- ***Application Programming Interface*, API**: conjunto de subrutinas, funciones y procedimientos (o métodos, en la programación orientada a objetos) que ofrece cierta biblioteca para ser utilizado por otro software como una capa de abstracción [Rou].
- ***Software Development Kit*, SDK**: conjunto de herramientas de desarrollo de software que permite a un desarrollador crear una aplicación informática para un sistema en concreto [Wikc].
- ***Front-End*, Frontend**: es la parte del software que interactúa con los usuarios [Wikb].
- ***Back-End*, Backend**: es la parte que procesa la entrada desde el front-end [Wikb].
- ***Representational State Transfer*, REST**: cualquier interfaz entre sistemas que utilice directamente HTTP para obtener datos o indicar la ejecución de operaciones sobre los datos[Wiki].
- ***MongoDB***: Base de datos documental no relacional [Wikd].
- ***Node.js***: Entorno de ejecución multiplataforma de JavaScript que utiliza un modelo asíncrono y dirigido por eventos[Wike].
- ***YAML***: es un formato de serialización de datos legible por humanos [Wikk].
- ***Oauth***: Es un estándar abierto que permite flujos simples de autorización para sitios web o aplicaciones informáticas. Se trata de un protocolo que permite autorización segura de una API de modo estándar y simple para aplicaciones de escritorio, móviles y web [Wikf].
- ***Docker Compose***: Docker-compose es una herramienta para definir y ejecutar aplicaciones Docker de contenedores múltiples. Docker compose utiliza un archivo

YAML para configurar los servicios de su aplicación. Luego, con un solo comando, crea e inicia todos los servicios desde su configuración [Docb].