E.T.S. de Ingeniería Industrial,
Informática y de Telecomunicación

Análisis históricos cuantitativos de Twitter

Grado en Ingeniería Informática

Trabajo Fin de Grado

Miguel Echenique Palacios

Alfredo Pina Calafi

Pamplona,18-06-2018

upna Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

Descripción Bibliográfica

Resumen

Este trabajo describe el proceso de investigar y desarrollar una solución que permita recuperar datos de la red social Twitter independientemente de su fecha de publicación y sin coste económico. Se parte de la identificación del problema por parte de la empresa navarra Tweet Binder. Se marcan unos objetivos, recuperar tweets independientemente de la fecha de publicación y hacerlo sin coste económico. A continuación se analiza el estado de la analítica de redes sociales y más concretamente en Twitter y se implementa una solución. Para terminar se despliega el resultado final en un servidor web y se prueba esa solución para obtener resultados y conclusiones del proceso.

Palabras clave

Twitter; analítica; histórico; API; web

Abstract

This Final degree Project describes the process of researching and developing a solution that allows recovering data from the social network Twitter regardless of its publication date and without economic cost. Firstly, this work begins with the identification of the problem, which is researched by the Navarre company Tweet Binder. In order to do that it sets goals, read tweets regardless of its publication date and without economic cost. Then, it Analyzes the social network's analytic status, specifically in Twitter and after that a solution is implemented. Eventually, the final result is displayed on a web server and that solution is tested in order to obtain results and conclusions from the process

Keywords

Twitter; Analyze; Historical; Api; Web

Índice

Introducción

Este trabajo y más concretamente su primera parte han recibido una gran influencia por parte de la empresa de analítica Tweet Binder, buena parte del trabajo se realizó durante mi estancia de prácticas en esta empresa, en lo que ha sido mi primer contacto con el mundo laboral.

¿Por qué analizar las RRSS?

En el mundo actual ya nadie duda del poder de influencia que tienen las redes sociales, comparable e incluso superior al que pueden tener otros medios de comunicación como la prensa escrita o la televisión. Sin embargo, pese a la existencia de herramientas para medir el impacto de periódicos (la tirada) o de programas de televisión (los medidores de audiencia), a menudo no resulta sencillo medir el impacto de las campañas que se realizan en redes sociales.

En muchos casos el impacto real de su esfuerzo en internet es una incógnita para la empresa. Esto acaba provocando que estas prioricen anunciarse en medios tradicionales, que aunque sean menos eficaces son más fácilmente medibles.

Si una empresa lanza una campaña va a querer saber cuánta gente la ha visto, que perfil es el de los usuarios que han participado, como ha evolucionado esta con el paso del tiempo, en qué zonas geográficas han tenido más éxito e incluso que ha dicho la audiencia acerca de la campaña. Realmente vemos que es mucho más rica la información que podemos obtener de una red social que la que proporcionan audiencias de TV o encuestas sobre otro tipo de campañas; además de que como ocurre con casi todo lo digital, el proceso se puede automatizar haciéndolo mucho más rápido.

Por todo lo expuesto anteriormente llama la atención que casi ninguna de las grandes redes sociales proporcione a sus usuarios herramientas realmente útiles de analítica, ofrecen herramientas poco potentes y que en muchos casos permiten analizar el rendimiento de perfiles o páginas de empresas pero no de lo que el resto de usuarios dicen de ellos o de otros en sus perfiles.

Twitter y su modelo de negocio

Twitter es una red social fundada en marzo de 2006. Su funcionamiento es simple, los usuarios pueden lanzar mensajes -“tweets”-, por lo general son texto plano aunque pueden ser multimedia y suscribirse a los “tweets” que publican otros usuarios -“follow”-. La red ha ido creciendo en funcionalidades a la vez que en usuarios; añadiendo conceptos como el “retweet“, compartir a tus “followers” el “tweet” original de otro usuario, y ya ha alcanzado los 330 millones de usuarios activos.

En lo concerniente a las analíticas Twitter es probablemente la red social más interesante de todas. En la mayor parte de las redes la configuración de privacidad es muy restrictiva y sólo permite a los usuarios consultar información de sus afiliados, lo que en Twitter serían los followers. Sin embargo Twitter es el salón con las cortinas corridas de las redes sociales, a menos que el usuario oculte su perfil -cosa que pocos hacen- cualquiera puede consultar sus actualizaciones, algo que la convierte en la red social idónea para hacer análisis cuantitativos.

La causa de por qué es Twitter tan poco restrictivo con la privacidad de sus usuarios debemos buscarla en su modelo de negocio. La mayor parte de las redes sociales como Facebook, basan la mayor parte de su monetización en publicidad abundante y perfiles profesionales que pagan por anunciarse, en otras como Linkedin los usuarios pueden desbloquear opciones pagando por ellas. Sin embargo en Twitter no hay funcionalidades de pago y apenas se potencia la capacidad de los usuarios de pagar para que los Tweets o los perfiles sean visibles para más gente.

Ante esto cabe preguntarse como gana dinero Twitter, la respuesta es sencilla, como veremos más adelante la red social vende los datos de sus usuarios a empresas externas que se dedican a analizarlos. La API de pago de Twitter es una de sus mayores fuentes de ingresos y para que esta funcione Twitter necesita que se pueda acceder a la mayor parte de los perfiles de sus usuarios, de ahí la configuración tan poco restrictiva de la privacidad. En el año 2016 el 13% de los ingresos de Twitter venían de esta fuente.

Reflexione sobre esto, Twitter es una granja inmensa de datos frescos, cada día se publican en torno a 500 millones de tweets por todo el mundo, se vierten opiniones sobre absolutamente todo y están disponibles para que cualquiera pueda acceder a ellas. Póngase por un segundo en el papel de una agencia de publicidad, en el de un director de una cadena de televisión o como encargado de una campaña electoral.. ¿No desea saber qué opinan miles o millones de personas de usted y de su competencia? ¿No está dispuesto a pagar por ello? De todo esto deriva el modelo de negocio de Twitter, de las empresas de analítica y también el objetivo de este trabajo.

Informes históricos en Twitter

Se ha explicado anteriormente que hoy en día nadie duda del poder de las redes sociales, pero esto no ha sido siempre así. Hasta hace 5 o 6 años apenas nadie se planteaba que tuviera interés analizar el impacto de una campaña en Twitter y tampoco había herramientas lo suficientemente potentes para hacerlo en profundidad.

Esto ha cambiado profundamente a día de hoy, ahora mismo todo se mide y todo se compara. Pero se pueden poner en contexto los datos de hoy si no se dispone de datos antiguos que comparar. Aunque obtener esos datos no resulta sencillo, por su naturaleza las redes sociales dan facilidades para acceder al contenido reciente, el que genera tráfico ahora. Sin embargo cuando se trata de acceder al contenido de años atrás, ese que es necesario para poder comparar datos y obtener feedback, se vuelve mucho más complicado y caro.

Twitter es el máximo exponente de esto dentro de las redes sociales, constantemente se está generando nueva información que se ordena de cara a los usuarios por orden cronológico; lo nuevo siempre es más importante, así siempre hay contenido que consumir. Esto hace que Twitter ponga muchas facilidades a las empresas para acceder a sus datos más recientes, incluso para hacerlo de forma gratuita. Todo esto se complica cuando queremos retroceder en el tiempo y el proceso se vuelve muy costoso económicamente.

Objetivos

De lo expuesto anteriormente surgen los tres objetivos que guiarán la realización de este trabajo:

- Desarrollar una herramienta que permita recoger información de Twitter desde el comienzo de la red social de forma automática
- Conseguir un coste que sea mínimo o incluso gratuito.
- Desarrollar y testear una interfaz para controlar la herramienta

Metodología

El proceso seguido para buscar una solución consta de varias fases. En primer lugar se analizarán todas las APIs que ofrece Twitter y como se realiza actualmente el acceso a los datos de la red social. En segundo se buscará una vía para obtener datos históricos de Twitter de forma gratuita. Posteriormente se desarrollará una herramienta web que permita automatizar el proceso anterior. Por último se diseñará una interfaz gráfica que permita controlar la herramienta y se probará esta interfaz y se analizarán los resultados obtenidos.

Estado del arte

Introducción: Twitter y el acceso a los datos

Nuestro objetivo es acceder a datos de twitter sin límite de tiempo, Twitter ofrece dos APIS a los desarrolladores: la gratuita y la de pago. La API de pago ha sido descartada para este proyecto por su elevado coste económico y debido a que exige registrarse como empresa en Twitter, cosa que escapa a los objetivos de este trabajo.

La Api gratuita de Twitter

Twitter provee a los desarrolladores que lo soliciten de acceso a su API REST gratuita. Una API REST o de transferencia de estado representacional es una arquitectura para el diseño de interfaces que se caracteriza por usar directamente el protocolo HTTP para el intercambio de datos en cualquier formato, aunque en los últimos años JSON se ha impuesto a todos los demás y es el utilizado por Twitter . REST se diferencia de los demás protocolos en:

- Es un protocolo sin estado, en cada mensaje está toda la información necesaria para comprender la petición y por tanto cliente y servidor no necesitan recordar el estado de la comunicación.
- Un conjunto de operaciones definidas, habitualmente POST, GET, PUT y DELETE.
- Una sintaxis universal para los recursos, cada recurso debe ser direccionable únicamente mediante su URI.

- El uso de hipermedios que permitan la navegación entre recursos simplemente siguiendo los enlaces.[1]

JSON (JavaScript Object Notation), es un formato de intercambio de datos ligero, basado en texto e independiente del lenguaje. Deriva del formato utilizado en ECMAScript (cuya implementación más conocida es JavaScript).[2]

JSON puede representar 4 tipo primitivos: cadena, número, booleano y nulo y dos estructuras de datos, objeto y vector. JSON se implementó como parte el ECMAScript pero posteriormente fue adoptado en más lenguajes por su ligereza y versatilidad reemplazando a otros formatos de datos como XML, más pesado que JSON. Se ha convertido en el estándar de la web y de la mayoría de APIs REST.

Los recursos de la API

Twitter ofrece en su web la documentación de su API. Está diseñada para poder hacer todo tipo de acciones pero en este trabajo solo la sección de los tweets resulta de interés. A continuación analizaremos los que pueden ser de utilidad.

Como se ha señalado anteriormente cada recurso que forme parte de una API REST es accesible mediante su propia URI. Todos los recursos de la API comienzan por https://api.twitter.com/1.1/ así que de ahora en adelante acortaremos la dirección y simplemente se mencionará la dirección relativa dentro de la ruta de la api, además todas las URIs que se van a presentar corresponden al método GET salvo que se especifique lo contrario.

El recurso más básico de la API gratuita de Twitter a la hora de buscar tweets es /search/tweets.json, simplemente devuelve una colección de tweets que encajan con la consulta especificada en los parámetros de la URL. Por ejemplo la consulta /search/tweets.json?q=unavarra devolverá los tweets que contengan la palabra unavarra. Es importante tener en cuenta varias cosas. En su API gratuita Twitter no garantiza el acceso a todos los tweets que cumplan la condición de la búsqueda, advierten claramente en la especificación de la API que no todos los tweets tienen por qué estar indexados. Por otra parte Twitter solo permite acceder a publicaciones de los últimos 7 días, no permite remontarse más atrás en el tiempo. Por lo tanto se descarta este recurso, ya que no permite ni siquiera recuperar tweets pasada una semana.

Hay muchos más recursos disponibles en la API, permiten acceder a usuarios, listas, mensajes... Debido a la imposibilidad de analizar uno a uno todos los que nos ofrece Twitter se explican a continuación los dos que emplearemos en nuestro proyecto.

Por una parte está /statuses/lookup.json, admite como parámetros id que es una lista de tweet ids separada por comas y diferentes parámetros relativos al formato de los tweets que no son de importancia para nuestro proyecto. Es decir, hacer un /statuses/lookup.json?id=982247277010939904,972119432167411712 nos devolverá un JSON que contendrá la información relativa a esos dos tweets.


```
[{
	"created_at": "Fri Apr 06 13:22:54 +0000 2018",
	"id": 982247277010939904,
	"id_str": "982247277010939904",
	"text": ""¿Qué es la tecnología?", una obra de divulgación para descubrir esta disciplina, publicada por la #UPNA y Editoria... https:\/\/t.co\/4TgxSqAXqW",
	"truncated": true,
	"entities": {
		"hashtags": [{
			"text": "UPNA",
			"indices": [
				99,
				104
			]
		}],
		"symbols": [],
		"user_mentions": [],
		"urls": [{
			"url": "https:\/\/t.co\/4TgxSqAXqW",
			"expanded_url": "https:\/\/twitter.com\/i\/web\/status\/982247277010939904",
			"display_url": "twitter.com\/i\/web\/status\/9...",
			"indices": [
				117,
				140
			]
		}]
	},
	"source": "<a href=\"http:\/\/twitter.com\" rel=\"nofollow\">Twitter Web Client<\/a>",
	"user": {
		"id": 124154741,
		"id_str": "124154741",
		"name": "UPNA - Universidad",
		"screen_name": "UNavarra",
		"location": "Pamplona, Navarra (España)",
		"description": "Cuenta oficial de la Universidad Pública de
```

```
Navarra (UPNA). Campus de Pamplona y Tudela. @campus_iberus
https:\/\/t.co\/Qe6EZd7Hve Tel: 948 169000",
            "url": "http:\/\/t.co\/5BZ2hIPx4j",
            "protected": false,
            "followers_count": 14188,
            "friends_count": 930,
            "listed_count": 294,
            "created_at": "Thu Mar 18 12:44:04 +0000 2010",
            "favourites_count": 566,
            "utc_offset": 7200,
            "time_zone": "Madrid",
            "geo_enabled": true,
            "verified": false,
            "statuses_count": 9465,
                                                    "profile_image_url":
"http:\/\/pbs.twimg.com\/profile_images\/950630075065618432\/brwgg64U_nor
mal.jpg"
        },
        "geo": null,
        "coordinates": null,
        "place": null,
        "contributors": null,
        "is_quote_status": false,
        "retweet_count": 0,
        "favorite_count": 1,
        "favorited": false,
        "retweeted": false,
        "possibly_sensitive": false,
        "lang": "es"
    },
    ...
]
```


Fig 1

Nota: La información de los tweets ha sido acortada para facilitar la lectura.

Por otro lado está /statuses/retweets/:id, :id corresponde al id del tweet del que queremos obtener sus retweets que también necesitamos para nuestros análisis, sin embargo este endpoint solo devuelve 100 de entre los más recientes retweets, es decir que si un tweet tiene 10000 retweets solo obtendremos 100. El formato será similar al anterior pero en vez de con tweets originales con retweets.[3]

La autenticación

Para poder acceder a estos recursos de la API Twitter nos exige autenticarnos mediante el protocolo OAUTH, un protocolo de autorización estandarizado por la Internet Engineering Task Force, entidad que regula la mayor parte de estándares de Internet (conocidos como RFC). Oauth fue desarrollado en el año 2006 para la red social Twitter pero pronto se vio la gran utilidad del protocolo y en el año 2007 se publicó una especificación abierta implementada por muchas compañías tecnológicas como Facebook, Microsoft, GitHub o Google.[4]

El objetivo de OAUTH es permitir al usuario autorizar a aplicaciones de terceros (como la nuestra) a acceder a recursos en su nombre manteniendo sus credenciales a salvo.Participan tres actores, el propietario de los recursos suele ser el usuario que permite acceder a recursos protegidos en su nombre, en nuestro caso buscar tweets o retweets. El cliente o consumidor es quien realiza el acceso a estos recursos, en este caso nuestra aplicación. Por último el proveedor es quien autoriza el acceso y sirve los recursos.

OAUTH emplea una identificación para la aplicación compuesta de una clave pública y otra privada y un token de acceso a los recursos del propietario (también con parte pública y privada) que combinado con estas claves permite el acceso a los recursos. Por lo tanto para poder acceder a los recursos de Twitter necesitamos una clave de acceso y al menos un token de usuario.

Más adelante se hablará sobre como obtener tokens de usuario para nuestra aplicación.

La búsqueda web

Twitter provee a sus usuarios de un motor de búsqueda, tanto en versión web como móvil. Este forma parte de la interfaz de Twitter y cualquier usuario de la red puede utilizarlo. El motor de búsqueda permite monitorizar usuarios, hashtags, tweets que contengan ciertas palabras, limitarlos por fecha de inicio y fin, por su autor…

El funcionamiento de este motor es sencillo, se introduce una consulta en la barra de búsqueda y aparecen los tweets ordenados por fecha de publicación -de más nuevo a más viejo-, uno tras otro al estilo de la “timeline” de un usuario.

Twitter ofrece al usuario muchísimos operadores de búsqueda que convierten al buscador en una de las herramientas más potentes que utilizan los usuarios avanzados de la red social, sin embargo apenas hay documentación sobre como emplearlo y muchos usuarios desconocen su potencial. Podemos limitar la búsqueda por términos usando AND y OR, por

fechas, por usuarios e incluso por localización espacial del usuario que publicó los Tweets.[5]

Este buscador presenta un problema claro; está pensado para ser empleado por usuarios, no por una herramienta automática.

Propuesta

Por todo lo expuesto anteriormente queda claro que no hay una manera clara de acceder de forma automática a los datos de Twitter que tengan un antigüedad mayor a 7-9 días con un coste económico asumible, esto es un periodo de tiempo realmente corto y dificulta muchísimo la realización de análisis. De esta carencia deriva el objetivo de este trabajo, conseguir la información que ofrece la API gratuita de Twitter pero evitando el límite de tiempo y sin asumir los costes de la API de pago.

Análisis del problema

Requisitos

Los requisitos para nuestro trabajo están claros, debe ser posible acceder a los tweets emitidos independientemente del momento de su publicación y además debe poder hacerse de manera gratuita y automática. Además se busca desarrollar una interfaz que permita controlar la herramienta y comprobar su usabilidad.

Solución propuesta

Hemos presentado anteriormente los recursos de la API gratuita de Twitter, gracias a ellos podemos obtener tweets originales sin fecha límite siempre y cuando tengamos el ID del tweet. Por tanto nuestra solución pasa por encontrar una forma de conseguir los IDs de los tweets que queremos analizar.

Por otra parte recordemos el motor de búsqueda de Twitter, indexa resultados sin límite de tiempo y podemos hacer consultas complejas con multitud de operadores. Se investiga la interfaz web del buscador para saber si es posible extraer el id de los tweets resultantes de una búsqueda.

Destacar que ahora ya no navegamos por una API Rest sino por un sitio web. El lenguaje más extendido para la presentación de webs es HTML(Hiper Text Markup Language), un

lenguaje de marcado que da formato a la mayor parte de webs del mundo y que los navegadores se encargan de renderizar para que el usuario pueda navegar.

Emplearemos las herramientas para desarrolladores del navegador web Firefox, concretamente usaremos el inspector de estilos y dom para analizar el código HTML de la web y buscar el identificador de los tweets dentro de él.

Revisamos el código HTML de la web y advertimos que el identificador de cada tweet aparece dentro del código de la página, por lo tanto uniendo el motor de búsqueda web con el endpoint de /statuses/lookup podremos obtener los tweets de nuestras búsquedas.

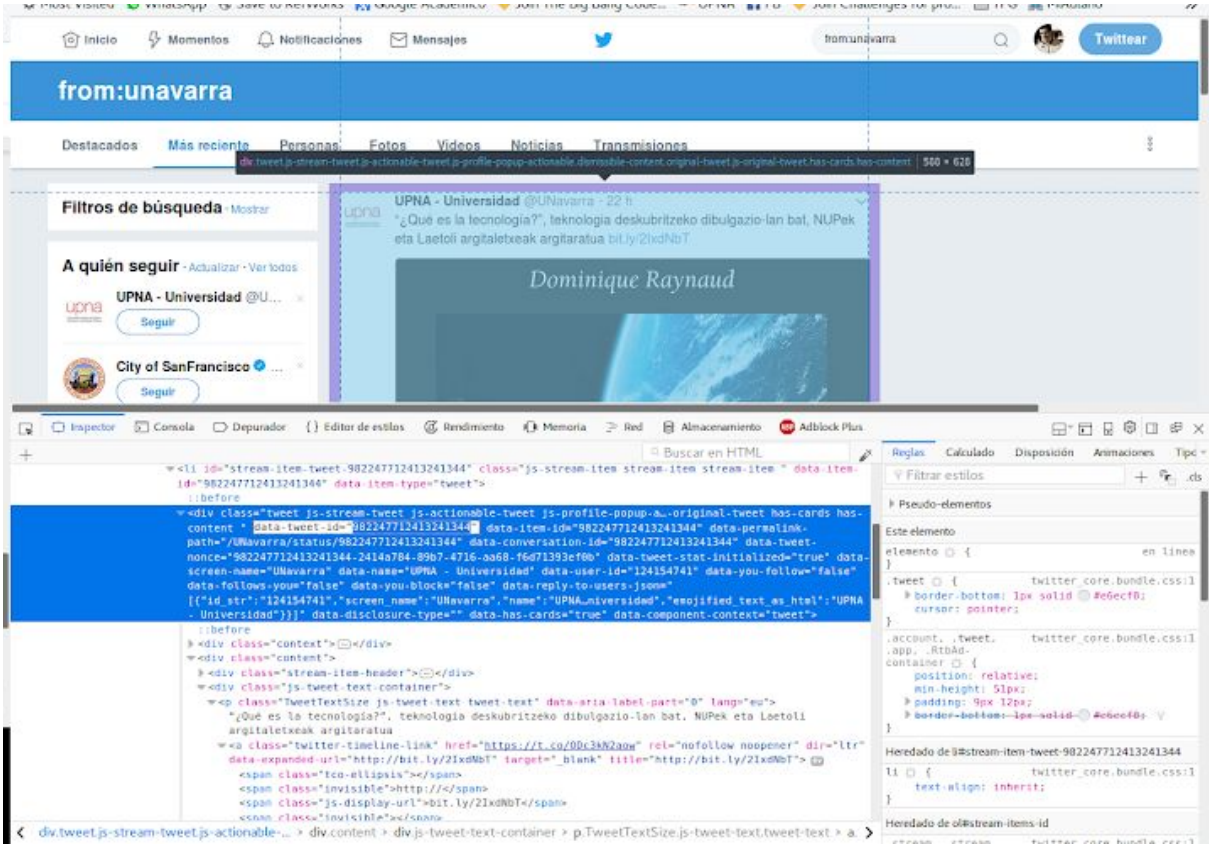


Fig 2

Aparece visible en la imagen el id del tweet seleccionado con la herramienta inspector de Firefox.

No obstante al cargar la web solo aparecen 20 tweets, el resto de los tweets de la búsqueda van cargándose de manera dinámica a medida que el usuario se desplaza hacia abajo en la página. Como el objetivo es automatizar el proceso no parece lógico depender de una persona que vaya deslizando por la web para cargar más tweets.

Así que empleando otra herramienta de firefox, el rastreador web, vemos todas las peticiones HTTP que se van haciendo a la API Rest interna de Twitter mientras se carga la página y conseguimos aislar la que nos interesa que tiene un formato:

GET https://twitter.com/i/search/timeline?f=tweets&vertical=default&q=upna AND ingenieria&src=tren&max_position=TWEET-976501367903698944-976501517434998785

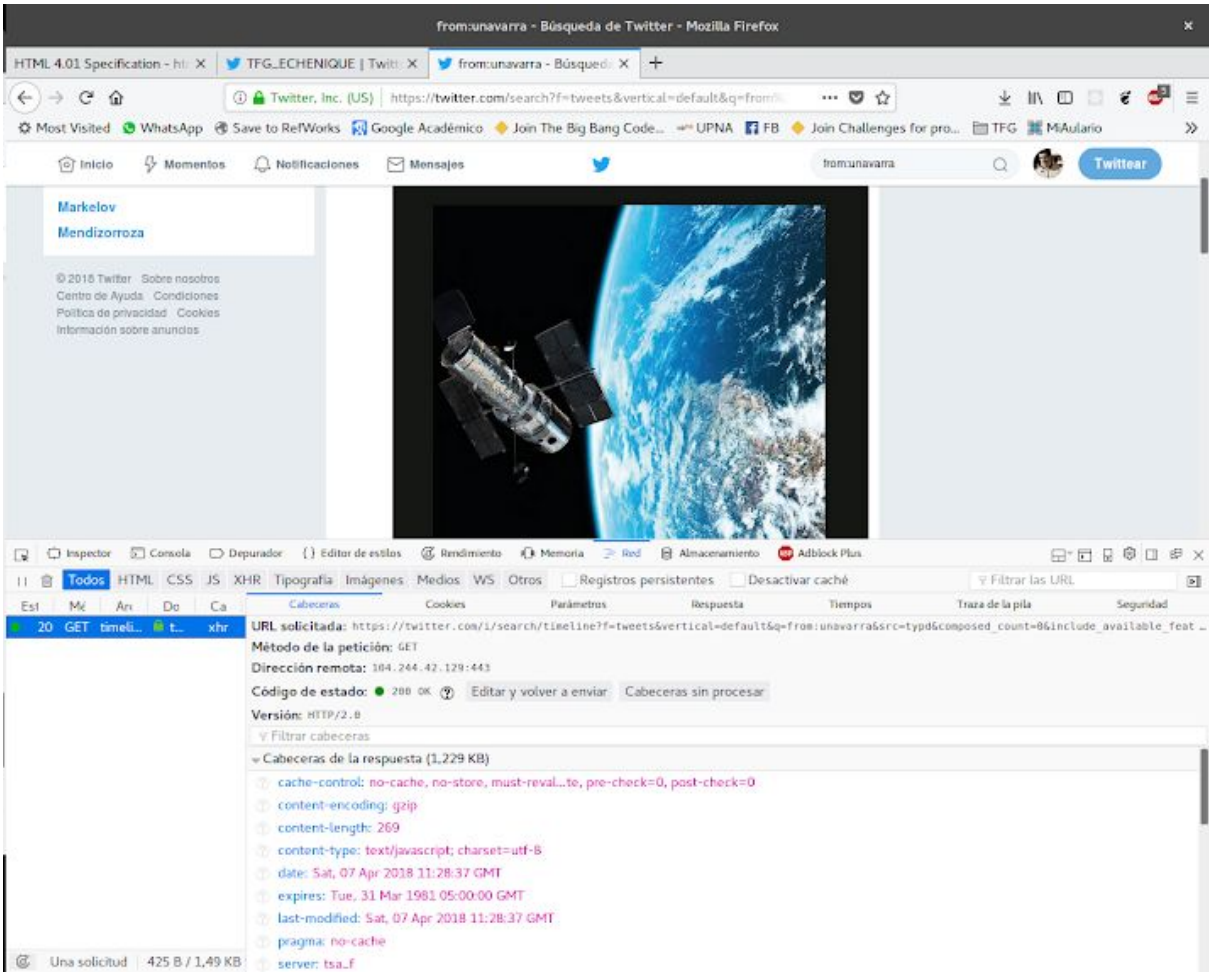


Fig 3

Rastreador de red de Firefox con la petición que se obtiene al pedir más “tweets” seleccionada.

A continuación analizamos la query:

- La f es el tipo de entidades que se presentan, si fuera users sería una búsqueda de usuarios por ejemplo.
- El vertical simplemente indica que los tweets deben cargarse uno encima de otro.
- La q es la búsqueda codificada, en este caso todos los tweets que contengan los términos upna e ingeniería.
- Max_position marca el último tweet que se había pedido anteriormente, Twitter devolverá los siguientes 20 tweets en orden temporal a ese. Como se ve no es un id en formato estándar y por tanto no servirá para hacer peticiones a la API gratuita, pero sí que podemos obtenerlo gracias a que está presente en el documento de respuesta de cada petición.

Si repetimos esta petición a Twitter obtenemos un documento JSON que parseado contiene MAX_POSITION (el parámetro indicado anteriormente), MIN_POSITION, que no es más

que el último tweet devuelto, items_html qué son los tweets listos para mostrar en pantalla en formato html y un campo booleano has_next_items, que es true si hay más tweets para cargar.

Por lo tanto para obtener todos los tweets simplemente tenemos que ir cambiando las peticiones para que cada vez MAX_POSITION sea el MIN_POSITION anterior e iremos obteniendo 20 nuevos tweets de los que podemos sacar su identificación.

Este será el funcionamiento de la herramienta, repetir la petición a twitter para ir obteniendo tweets, analizarlos para obtener su id y agruparlos para hacer peticiones a la API gratuita de Twitter de 100 en 100 hasta completar la búsqueda.

Diseño de la solución

El desarrollo de la herramienta

Anteriormente se expuso el método a utilizar para obtener tweets sin límite de tiempo, a continuación se explicará el proceso seguido para implementar una solución que permita automatizar el proceso.

Tecnologías empleadas

Node

Node es un entorno de ejecución de JavaScript orientado a eventos asíncronos, diseñado para construir aplicaciones en red escalables. A pesar de que Node está enfocado a construir servidores web es la herramienta ideal para nuestro proyecto gracias a:

- La facilidad para parsear los objetos en formato JSON que devuelve la API de Twitter ya que son el formato nativo de Javascript.
- Su enfoque hacia HTTP lo hace ideal cuando se trata de atender peticiones como se hace cuando se crean servidores como se suele hacer en Node pero también en nuestra aplicación que va a hacer multitud de peticiones tanto a la API de twitter como al motor de búsqueda.
- Está orientado a eventos asíncronos lo que mejora mucho el rendimiento en un software como el nuestro que la mayor parte del tiempo estará esperando a recibir la respuesta a una petición HTTP.[6]

JavaScript

Javascript es un lenguaje de programación, es una implementación del estándar ECMAScript cuyo desarrollo comenzó en el año 1996. Inicialmente fue diseñado para ser utilizado principalmente en el lado cliente para manipular el contenido de las páginas web de forma dinámica pero gracias a Node se puede emplear en el lado del servidor. JavaScript ha ido ganando en funcionalidades con el paso del tiempo hasta situarse como uno de los lenguajes de programación más populares del mundo.

Javascript emplea una sintaxis similar a la de C, aunque también adopta nombres y convenciones de Java. Actualmente JavaScript se define como un lenguaje orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.[8]

NPM

Npm es uno de los principales gestores de paquetes de Node, con él podemos añadir paquetes de terceros a nuestro proyecto de manera muy sencilla, guardando además la información de qué paquetes y versiones hemos instalado en un fichero llamado package.json, que luego permitirá volver a instalar las librerías en cualquier entorno de ejecución o desarrollo.

A continuación se exponen los principales paquetes de NPM que vamos a utilizar en nuestro proyecto.

Request

Request es un módulo diseñado para facilitar las peticiones de HTTP, su uso es muy simple y a la vez es muy potente, admite autenticación por OAUTH, necesaria para la API de Twitter, parsea de forma nativa los documentos en JSON y tiene más funcionalidades que no serán necesarias en este proyecto como la creación de streams.

Himalaya

Himalaya es un parser de HTML a JSON, esto nos permitirá tomar el HTML que devuelve el motor de búsqueda de Twitter con los tweets y convertirlo a JSON para manipularlos en node y extraer sus ids.

Async

Async es una librería desarrollada por Caolan que permite controlar de manera sencilla el flujo de ejecución de Javascript, gracias a ella evitamos que el uso de callbacks en nuestro programa se descontrole ya que nos proporciona diferentes funciones para gestionar la asincronía de Node.

Express

Expressjs es un framework web minimalista utilizado junto a Node para la creación de servidores web, nosotros lo emplearemos para exponer APIs REST en varios servicios.

SQS

Amazon Simple Queue Service (SQS) es un servicio de colas de mensajes diseñado para enviar, almacenar y recibir mensajes entre componentes de software de cualquier volumen garantizando no perder mensajes ni tener que acceder a otros servicios para lograr una disponibilidad permanente.

En nuestro caso tiene mucha utilidad, de esta forma podemos tener uno o unos pocos productores que reciban los trabajos que debemos procesar y más consumidores que lean de la cola de SQS y procesen estos trabajos.

MongoDB

Mongodb es una base de datos noSQL muy utilizada en entornos de desarrollo con Node. Ya que permite cambiar los esquemas de los documentos rápidamente, algo que en el caso de nuestra aplicación es probable que ocurra, ya que no todos los tweets tienen el mismo esquema y Twitter no se compromete a mantenerlo.

Además Mongo almacena documentos en formato JSON, lo que nos permite insertar directamente los objetos con los que tanto la Api de Twitter como nuestro lenguaje (JavaScript) trabajan.

En nuestro caso usaremos la base de datos en los servicios que trabajan con Ids ya que no podemos abusar de la capacidad de SQS para encolar tantos tweets como necesitamos, simplemente los almacenaremos en Mongo y los iremos leyendo y marcando como procesados.

Docker

Docker es un software diseñado para poder utilizar el mismo entorno de ejecución de forma independiente a la máquina que estemos empleando. Docker básicamente crea un contenedor de software en la máquina que virtualiza el sistema operativo permitiendo aislar el contenedor dentro del kernel de Linux, evitando la necesidad de crear máquinas virtuales. En nuestro proyecto se usará para ejecutar tanto los servicios como SQS y mongo en el entorno de desarrollo.

Conseguir acceso a la API gratuita

Ya se ha mencionado antes que se pretende emplear la API gratuita de Twitter para desarrollar la información. Conseguir acceso es algo realmente sencillo y rápido. Nos dirigimos a https://apps.twitter.com/, nos logueamos con una cuenta de Twitter que debe haber verificado su teléfono y pulsamos en el botón Create New App. Completamos los datos que se nos piden y pulsamos en Create your Twitter application.

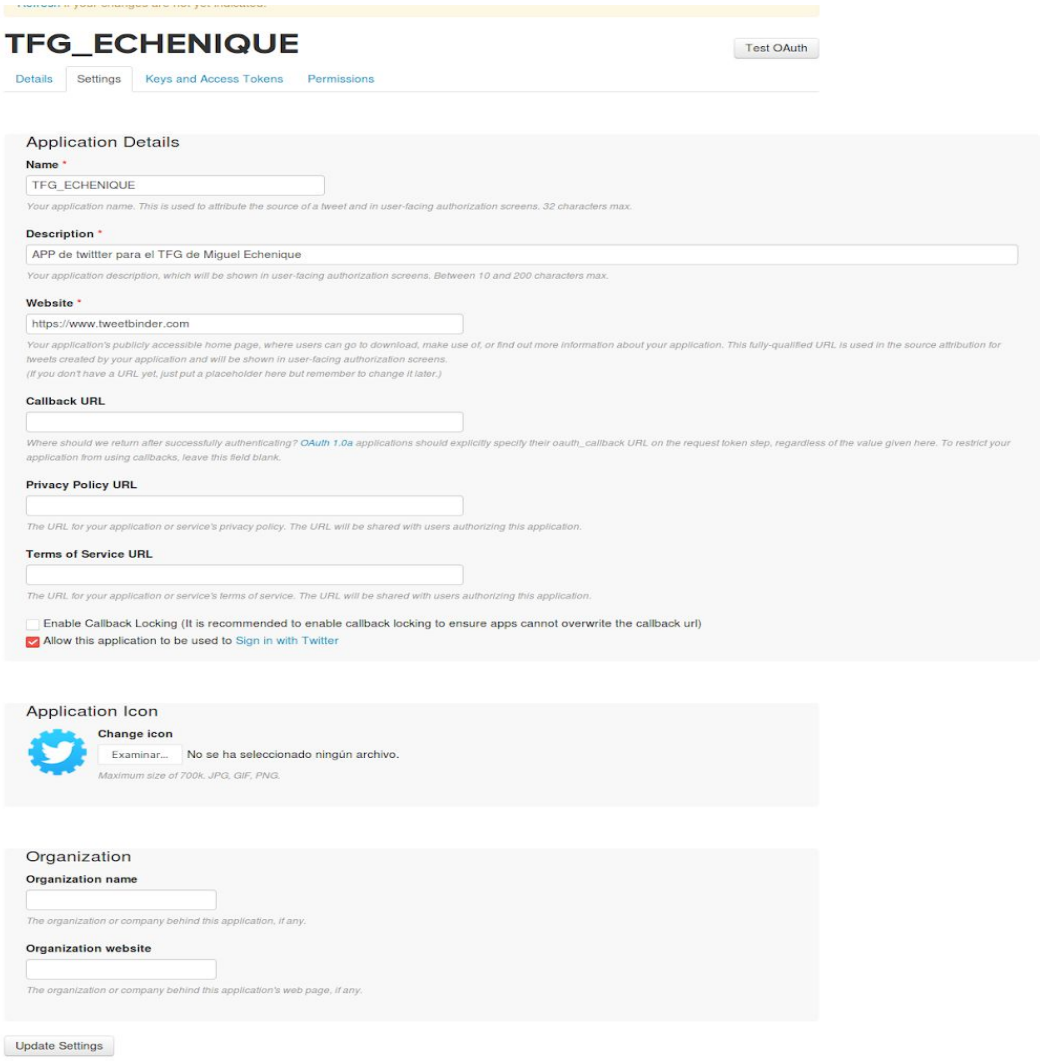


Fig 4

Para poder hacer peticiones a la API de Twitter tan solo necesitamos utilizar el token del usuario junto a la clave de la aplicación (y sus correspondientes secretos). Obviamente las claves no van a aparecer en este documento por seguridad.

Una vez tenemos el token de usuario y nuestra clave ya podemos acceder a los recursos de la API gratuita, sin embargo debemos tener en cuenta que la API tiene un límite de 150 peticiones por token cada 15 minutos, si queremos poner nuestra APP en producción necesitaremos más tokens de usuario, sin embargo mientras estemos desarrollando con un token será suficiente.

La arquitectura de la solución

Como se ha explicado anteriormente el flujo básico de nuestro programa consiste en hacer peticiones al motor de búsqueda para obtener los ids de los tweets que queremos analizar y posteriormente preguntar a la API gratuita de twitter por estos tweets. Para maximizar la modularidad de nuestra aplicación y permitir su desarrollo, actualización y escalabilidad de la forma más rápida y eficiente posible se ha optado por dividirla en varios microservicios que serán:

- Twitter-web-scraper-producer: Este servicio expondrá una interfaz de HTTP a la que se podrán añadir las búsquedas a realizar y las añadirá a una cola de SQS, además permitirá listar las búsquedas, tanto pendientes como completadas y cancelarlas.
- Twitter-web-scraper-consumer: Irá tomando trabajos de la cola del SQS y realizando las búsquedas parseando los Ids de los tweets y añadiéndolos al siguiente servicio.
- Twitter-statuses-producer: Expone una interfaz similar a la anterior, en este caso permite añadir Ids a una colección de mongo que el siguiente servicio procesará.
- Twitter-statuses-consumer: Va tomando ids de la colección de mongo y agrupándolos para pedir su información a twitter de 100 en 100 y luego pedir sus retweets en el caso de que los haya. Si son más de 100 además encola un nuevo trabajo en el primer servicio para obtener el máximo número de retweets posible. Cuando tiene la información de los tweets la envía a kafka.

Nota: Los servicios usan una librería común llamada shared-service. El objetivo era convertirla en un paquete de NPM pero debido al coste económico que ello conlleva se ha añadido como una carpeta llamada extra dentro de cada servicio.

El flujo

A grandes rasgos, el flujo que sigue una búsqueda desde su inicio hasta su finalización es el siguiente:

Primero la búsqueda se añade mediante una petición HTTP POST / al interfaz del twitter-web-scraper-producer. Esta petición contendrá la query de la búsqueda y el identificador del informe al que pertenecen estos tweets dentro de la base de datos de Tweet Binder.

Cuando el twitter-web-scraper-producer recibe la petición encola la búsqueda en el SQS y además la registra en una colección de mongo, esto lo hacemos por dos razones, la primera es tener un registro del estado de las búsquedas que poder consultar y la segunda es poder marcar búsquedas como canceladas en el caso de que el usuario se arrepienta, cosa que SQS no nos permite hacer.

Una vez la búsqueda está en SQS en algún momento será recibida por el twitter-web-scraper-consumer que comprobará en mongo que la búsqueda esté pendiente y comenzará a procesarla. Hará peticiones al motor de búsqueda de Twitter utilizando la query que ha recibido e irá parseando los ids de los tweets. Como no sabemos cuántos tweets va a tener nuestra búsqueda hasta que no la procesamos completamente, cada vez que tenga 500 identificadores listos hará una petición HTTP al servicio de

twitter-statuses-producer para encolar nuestros tweets y no tener que guardarlos todos en memoria.

Cuando el twitter-statuses-producer vaya recibiendo tweets mediante HTTP simplemente los introducirá en la colección de mongo.

Por último el twitter-statuses-consumer irá leyendo de la colección de mongo y pidiendo los tweets a la API de Twitter para emitirlos a Kafka. Además si el tweet tiene retweets también los pedirá y si son más de 100 tweets añadirá un nuevo trabajo para obtener el mayor número de retweets posible.

Los servicios

A continuación se procederá a ampliar las explicaciones sobre las funcionalidades de cada servicio web.

Twitter-Web-Scraper-Producer

Este servicio simplemente expone al exterior una API Rest con cuatro recursos:

- GET /
    - Devuelve la información de todos los trabajos en la cola. Simplemente el servicio lanza una consulta a Mongo para obtener la información de todos los trabajos y la devuelve como Array en un JSON.
- POST /:reportId
    - Crea un nuevo trabajo. Para ello registra la búsqueda con estado pendiente en Mongo y la inserta en la cola del SQS para que en algún momento el consumidor la lea y comience a procesarla.
- PUT /:reportId/cancel
    - Cancela un trabajo pendiente. Si la búsqueda no ha sido marcada como in-progress en mongo simplemente la cancela, si la búsqueda está en marcha además envía un mensaje al twitter-web-scraper-consumer para detener la búsqueda. POST /:reportId/retweets/:parentTweet
    - Crea un nuevo trabajo para obtener los retweets del tweet especificado. Se lanzará una query como la explicada antes (filter:nativeretweets + el texto del tweet original) y además se incluye el identificador del tweet original para evitar que si el texto es muy corto aparezcan tweets no deseados que también lo contengan. Por lo demás el proceso es igual que el del post /:reportId

Twitter-Web-Scraper-Consumer

Este servicio va procesando mensajes del SQS, cada vez que recibe uno realiza el siguiente proceso:

1. Toma el reportId (y el identificador del tweet original si son retweets) y va a Mongo a leer el trabajo.
2. Lee la información del informe para verificar que esté pendiente y su query. Marca el informe y la búsqueda como processing e in-progress.
3. Comienza a Scrapear la búsqueda de Tweets. Ese proceso se analizará más abajo.
4. Al terminar marca el report y la búsqueda como procesados o guarda el error en caso de que la búsqueda haya fallado.
5. Se marca el mensaje de SQS como error (volverá a ser encolado) o como procesado (se elimina)

Para obtener los Ids de los tweets se usa una función recursiva que funciona de la siguiente forma:

1. Hace la petición a Twitter explicada en el capítulo anterior para cargar un batch de tweets, si es la primera no indica el MAX_ID.
2. Una vez recibido el batch en formato JSON toma el maxId y los tweets en formato HTML que son parseados usando Himalaya para buscar el tweet o retweetId.
3. Si lleva 500 o más Tweets envía los tweets al statuses-producer y los borra de la memoria.
4. Si ha encontrado Tweets volver al paso uno. Si no los encuentra vuelve a intentar cargar más tweets con la misma petición hasta 10 veces para dar tiempo a Twitter a indexarlos y mostrarlos, posteriormente sale.

Twitter-Statuses-Producer

Ofrece una API Rest similar a la del Twitter-Web-Scraper-Producer, sus recursos son:

- PUT /add
    - Añadir nuevos statuses (tweets) a Mongo para que el consumer los procese.
- PUT /delete
    - Borra los statuses indicados en el cuerpo de la petición
- PUT /delete/all
    - Borra todos los statuses pendientes
- PUT /delete/:reportId
    - Borra todos los statuses correspondientes a ese reportId

Twitter-Statuses-Consumer

1. Cada cierto tiempo, actualmente 5 segundos pero es configurable, se levanta y toma hasta 500 de los statuses pendientes en grupos de 100.
2. Los marca para que ninguna otra instancia de este servicio los procese a la vez.
3. Hace la petición al recurso de twitter /statuses/lookup.json?id= y añade los ids separados por comas
4. Envía la información de los statuses al streaming de Kafka donde serán procesados.
5. Para cada status si tenía retweets se pide a Twitter la información de estos en el recurso /statuses/retweets/:idTweetOriginal.
    a. También se envían estos retweets al streaming de Kafka

6. Si el status tenía más de 100 retweets se añade un nuevo trabajo en el recurso del twitter-web-scraper-producer POST /:reportId/retweets/:idTweetOriginal
7. Se borran de mongo los statuses ya procesados.

Desarrollo fuera de Tweet Binder

Hasta aquí los servicios involucrados en la búsqueda y la parte que se realizó en la empresa Tweet Binder, con el fin de poder mostrar un proyecto completo y terminado se ha decidido continuar hasta crear una plataforma de analítica de Twitter muy básica pero eso sí, completa desde el registro hasta la visualización de un pequeño informe.

Básicamente se ha añadido una interfaz web que se explicará más adelante, un servicio llamado API que encapsula las llamadas a todos los demás, un servicio para el login con Twitter (authorization) y otro que genera los informes (stats).

Otros servicios

Api

La api es el servicio que encapsula las llamadas a todos los demás, es el único servicio al que llamará la interfaz web y por tanto el único en el que tendremos que autenticar al usuario. La api simplemente recibe llamadas a sus recursos y se comunica con el resto de servicios para obtener una respuesta.

Authorization

El proceso de autorización en OAUTH es un estándar empleado en muchísimas webs mediante el cual el usuario autoriza a una aplicación de terceros a acceder a sus datos en su nombre manteniendo a salvo sus credenciales. A continuación se adjunta un ejemplo tomado de la web de Spotify pero es idéntico para Twitter.

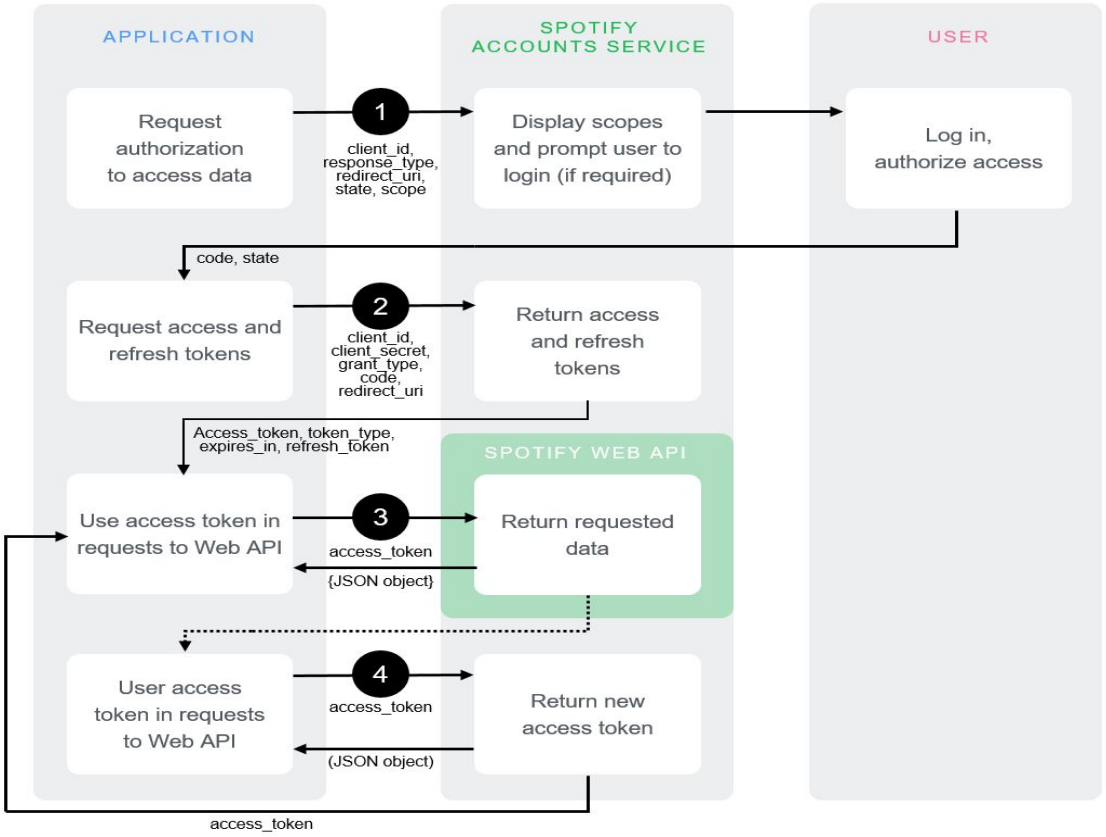


Fig 5 [7]

Como se ve el flujo consiste en que la aplicación solicite con las claves que la identifican una url que mostrar al usuario para redirigirlo a la web que debe autorizar el acceso a sus datos. Una vez el usuario nos autoriza es dirigido a nuestro sitio donde nos proporcionará unos tokens temporales que junto con nuestras claves podremos intercambiar por los definitivos.

Por tanto el funcionamiento de la Autorización es:

Un endpoint que hace una petición a Twitter y obtiene un token para redirigir al usuario a Twitter y un secreto que guardamos en mongo junto al token.

Otro endpoint que es la url a la que Twitter redirige a los usuarios cuando nos autorizan a acceder a los datos, esta hace una petición del servidor a Twitter uniendo los datos del usuario, nuestras claves y el secreto que hemos guardado. Twitter devuelve las claves de acceso que verificamos haciendo otra petición para obtener así el perfil del usuario en Twitter y guardarlo en nuestra base de datos junto a sus tokens de OAuth.

Stats

Recordamos que el Twitter-Statuses-Consumer emitía los tweets a un streaming de Kafka, por las limitaciones económicas y materiales en esta versión de la herramienta se ha optado

por eliminar Kafka y en vez de eso introducir los tweets en una colección de mongo llamada como el Id del informe que se está generando.

El servicio de las stats simplemente hace consultas contra esa colección y calcula las estadísticas del informe que mostramos en la interfaz web, a continuación se adjunta un ejemplo de respuesta.


```
{
    "totalTweets": 740,
    "mostActive": [
       {
             "_id": "CAOsasuna",
             "name": "CAOsasuna",
                                                          "profileImage":
"http://pbs.twimg.com/profile_images/974647941242769409/aS7x_wu__normal.j
pg",
             "backgroundImage": null,
             "followers": 214627,
             "tweets": 44
       }
    ],
    "mostPopular": [
       {
             "_id": "Cordobacfsad",
             "name": "Cordobacfsad",
                                                          "profileImage":
"http://pbs.twimg.com/profile_images/921246083325538304/40O06Bu3_normal.j
pg",
             "backgroundImage": null,
             "followers": 248764,
             "tweets": 1
       },
    ],
    "mostPopularTweets": [
       {
             "_id": "982941855359946752",
             "createdAt": "Sun Apr 08 11:22:55 +0000 2018",
             "text": "🎉 #SportingB\n\n🟦 FINAL en Tajonar 🆚 @CAOsasuna
B ➡ 0-1\n\n⚽ Cayarga 6'\n\n🔴⚪ ¡¡¡Enhorabuena guajes !!! 🔴⚪
https://t.co/dFFjYvKo94",
             "user": {
                   "_id": "342171958",
                   "name": "RealSporting",
                   "followers": 199474,
                   "following": 125,
```

```
                                                                "profileImage":
"http://pbs.twimg.com/profile_images/980867640301367297/NykntB2h_normal.j
pg",
                    "backgroundImage": null
            },
            "retweets": 139,
            "favs": 252,
            "isRetweet": false
        },
    ]
}
```


Nota: Se han reducido los arrays a un solo elemento para hacer el documento más legible y sencillo.

La interfaz web

Se ha diseñado una interfaz web muy sencilla que permita completar un proceso de búsqueda completo, es decir, enviar una búsqueda, ver el listado con todas las búsquedas y su estado y mostrar el resultado de un informe, básicamente el total de tweets, los tweets más compartidos (retweeteados), los autores con más followers y los autores que más veces aparecen en la búsqueda.

La web se ha desarrollando usando HTML, CSS y Javascript sin emplear ningún framework ya que por la escasa complejidad del desarrollo se descartó por lo que condicionan el desarrollo y el peso extra que suponen al sitio web.

Tecnologías

- HTML: Es un lenguaje de marcado ampliamente extendido en internet, cualquier navegador puede leerlo y actualmente es el estándar en el diseño de páginas web.
- CSS: Es un lenguaje de diseño gráfico para definir la presentación de un documento estructurado en un lenguaje de marcado, permitiendo así separar el formato del contenido y reutilizar los formatos una y otra vez.
- Javascript: El mismo lenguaje empleado en el resto del proyecto. Casi todos los sitios webs modernos lo encapsulan en un framework pero se ha decidido no hacerlo para evitar la complejidad adicional que esto requiere. Si que se emplea la librería jQuery, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos… Además de la librería Axios para hacer peticiones por HTTP.

Para facilitar el desarrollo del sitio se ha partido de una plantilla de Dashboard con licencia MIT tomada de la siguiente URL: https://www.creative-tim.com/product/light-bootstrap-dashboard

La plantilla emplea bootstrap que es un framework de html y css que contiene plantillas para facilitar el diseño y la maquetación, especialmente cuando se pretende que el diseño de la web se adapte a diferentes tipos de pantalla [9]. Fue desarrollado por Twitter aunque es OpenSource, llegó a ser el proyecto más popular de GitHub y lo emplean la mayor parte de sitios web actuales.

El sitio

El sitio está compuesto por 3 páginas. El listado de informes, el formulario de creación de informes y la página de informe propiamente dicha. Además se ha implementado autenticación para que solo los usuarios logueados puedan enviar búsquedas y los usuarios puedan acceder solo a su listado de informes (sin embargo la página de cada informe es pública para que los usuarios puedan compartirlos con sus clientes u otros interesados).

La autenticación

Para la autenticación se han empleado JWT o JSON Web Tokens, es un estándar abierto (RFC 7519) que define una forma compacta y autocontenida para transmitir información segura entre dos partes. La información puede ser verificada porque está digitalmente firmada mediante un secreto (como en este caso) o mediante un par de claves RSA.[10]

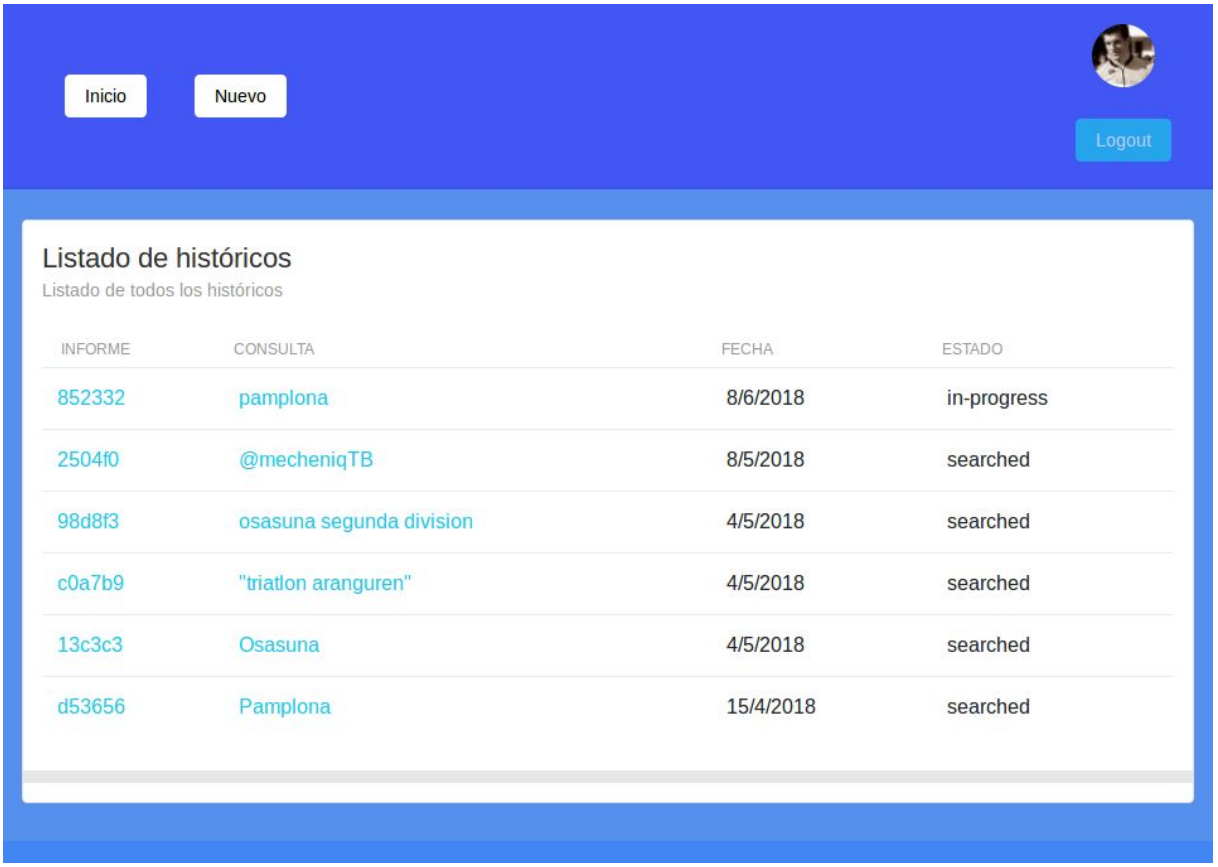
Además de servir para autenticar al usuario, si se dispone del secreto, los JWTs firmados pueden contener información del usuario accesible a cualquiera que lea el token. Esto permite acceder a su información en el cliente por ejemplo para cargar el nombre del usuario o su foto de perfil. Debemos tener en cuenta que la información que coloquemos en un JWT es visible a cualquiera que lea el token, por lo tanto no debería haber información confidencial en él.

El proceso de autenticación ya ha sido explicado en el servicio de la autorización, una vez el proceso se termina el servicio genera un JWT que guardamos en el navegador del usuario y enviamos cada vez que hace una petición. Este token es verificado y si el usuario está tratando de acceder a un recurso que le pertenece se le autoriza el acceso.

En el logout simplemente se borra el token del navegador del usuario y es forzado a recargar la página. Para aumentar la seguridad del sitio se debería invalidar el token en ese momento pero por el momento se ha considerado que al no haber datos confidenciales no era necesario el desarrollo de esa funcionalidad en un MVP (*Minimum Viable Product*).

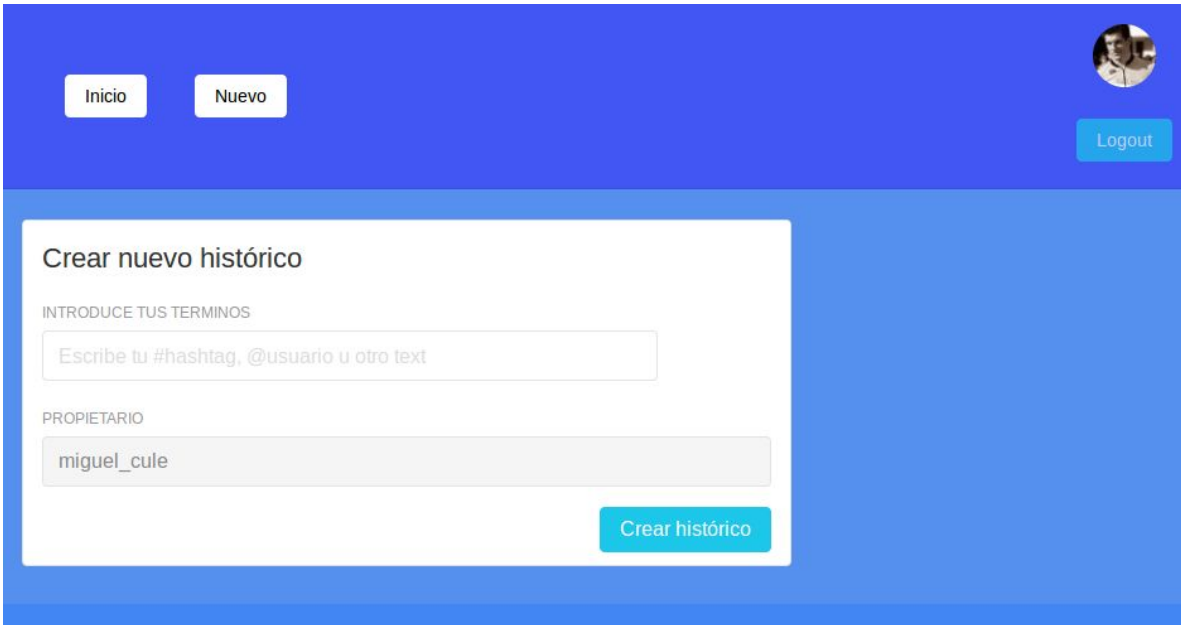
El listado de informes

El listado de informes es la página que aparece al cargar el sitio web. Simplemente hace una petición a la API a GET /searchs y esta, si el usuario está autenticado, le devuelve todos sus informes. Una vez llegan los informes se renderizan en una tabla en la que el usuario puede verlos todos y acceder al que más le interese. Este proceso se repite cada 5 segundos para cargar nuevos informes y también sus estados.



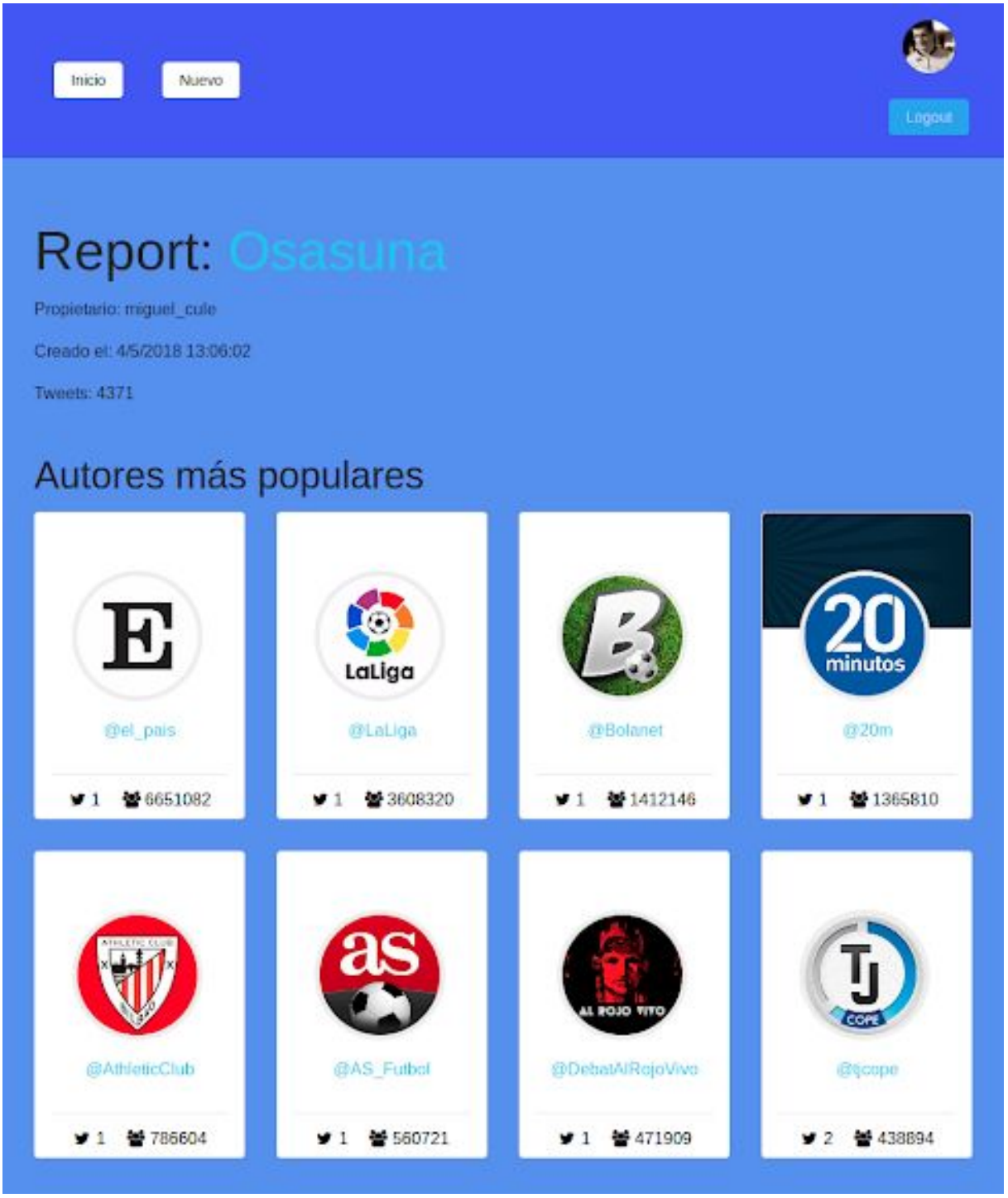
Enviar nuevo informe

Simplemente aparece un formulario en el que el usuario puede introducir su consulta tal y como la pondría en la búsqueda avanzada de Twitter y debajo un campo no editable con el nombre del usuario que genera el informe. Si el usuario no está autenticado este campo está vacío y el botón de enviar informe se sustituye por uno de Sign in With Twitter. Cuando el informe se envía se muestra una alerta al usuario para que vea que el informe ya está en marcha.



La página del informe

Toma la identificación del report de la Url y pide la información del informe a la API GET/reports/:reportId . Una vez está cargada simplemente la renderiza mostrando el propietario, el total de Tweets, los Tweets más populares, los autores con más seguidores y los autores que más aparecen.


Inicio
Nuevo
Logout
Report: Osasuna
Propietario: miguel_cule
Creado el: 4/5/2018 13:06:02
Tweets: 4371
Autores más populares
@el_pais
1 6651082
@LaLiga
1 3608320
@Bolanet
1 1412146
@20m
1 1365810
@AthleticClub
1 786604
@AS_Futbol
1 560721
@DebatAlRojoVivo
1 471909
@tjcope
2 438894

El despliegue

La puesta en producción de un total de 7 servicios, una interfaz web, una base de datos y un servicio de mensajes no es algo trivial. No es viable estar desplegando cada cambio en el código de forma manual pues el proceso sería extremadamente lento y además se pretendía que el despliegue de toda nuestra infraestructura fuera gratuito. Al final las opciones escogidas han sido:

AWS SQS

Una de las razones para elegir Simple Queue Service es que Amazon ofrece una capa de uso gratuita muy generosa para esta herramienta. El primer millón de peticiones mensual a esta herramienta es gratuito y eso es mucho más de lo que va a necesitar nuestro servicio.

Para el despliegue es necesario crear una cuenta en Amazon Web Services y acceder a la consola de desarrolladores. En la pestaña del servicio SQS creamos una nueva cola. La funcionalidad de SQS ofrece muchas opciones añadidas como la posibilidad de enviar los mensajes procesados con error a otra cola o la de establecer el tiempo máximo a partir del cual el mensaje se considera como erróneo aunque no haya respuesta y otro consumidor puede leerlo.

MongoDB Atlas

La propia Mongo ofrece un servicio de hosting de sus bases de datos. El proceso de creación es realmente sencillo y ofrecen una versión gratuita con un cluster formado por tres nodos y una velocidad de acceso realmente buena. El tamaño máximo es de 512 mb pero es más que suficiente para este proyecto.

Para crear un cluster simplemente se introduce el nombre del cluster y un usuario y contraseña para acceder, además de escoger la localización (en nuestro caso Frankfurt por proximidad geográfica). Al crear el cluster se anota la url de acceso que usarán los servicios.

Heroku

No es viable estar desplegando cada cambio en el código de los servicios, es algo que alargaría demasiado el proceso y no dejaría crecer la aplicación. Por tanto se ha buscado una herramienta que permita la integración continua (aunque todavía no tengamos un

entorno de integración y unos tests desarrollados) y el despliegue desde el repositorio de código (GitHub).

La elección ha sido Heroku que cumple todas estas condiciones y además ofrece hasta mil horas gratuitas de computación al mes si la cuenta es verificada con una tarjeta de crédito, algo que no conlleva un coste económico.

El proceso para crear cada servicio es tan sencillo como introducir el nombre de la aplicación y escoger la forma de despliegue que en este caso será desde un repositorio de Github.El usuario escoge el repositorio y la rama desde los que quiere que se haga el despliegue. Se activa el despliegue automático y a partir de ahora cada vez que haya un cambio en el código de esa rama el código del nodo en heroku se actualiza automáticamente.

Además Heroku permite configurar variables de entorno para la aplicación que serán empleadas para almacenar claves y direcciones url de acceso a otros servicios o estructuras de datos como Mongo o SQS.

La interfaz

La interfaz está formada solo por contenido estático, los archivos que se sirven del servidor no cambian. Por tanto se ha optado por crear un servidor de express con nodejs que sirva estos ficheros y darla de alta en Heroku como un servicio más. El funcionamiento es rápido y es la forma más simple de hacerlo ya que empleamos toda la estructura utilizada en los otros servicios. La web puede encontrarse en https://dashboardtfg.herokuapp.com/, sin embargo tenga en cuenta que al ser un servidor gratuito las primeras peticiones son muy lentas (pueden llegar a tardar hasta 30 segundos), tras la primera llamada a cada servicio el flujo se normaliza.

Resultados

El testeo de la interfaz

Se ha decidido testear la usabilidad de la interfaz mediante un test SUS (System Usability Scale), es un test creado por John Brooke en 1986 para medir una gran variedad de productos y servicios tanto hardware como software.Consiste en un cuestionario de 10 preguntas, cada una con 5 opciones. Está ampliamente aceptado como un test válido para diferenciar sistemas usables y no usables.[11] Se ha escogido por ser un test sencillo de aplicar y que no requiere de mucho tiempo ni esfuerzo a los voluntarios que lo realicen.

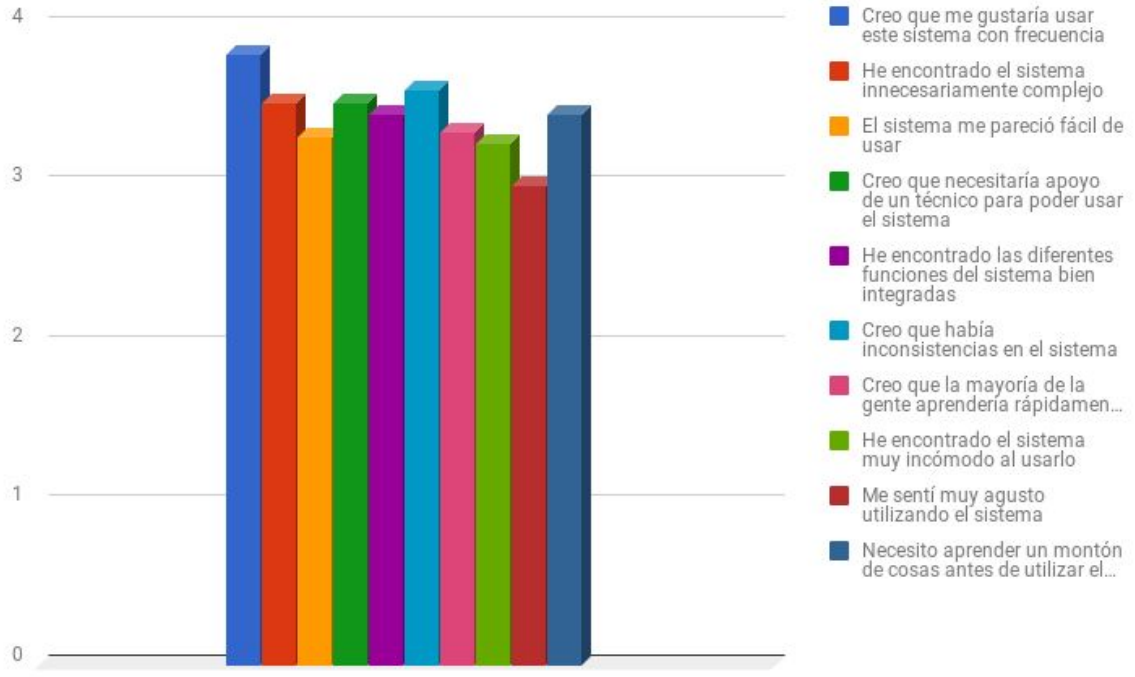
El test SUS consta de las siguientes preguntas:

- Creo que me gustaría usar este sistema con frecuencia
- He encontrado el sistema innecesariamente complejo
- El sistema me pareció fácil de usar
- Creo que necesitaría apoyo de un técnico para poder usar el sistema
- He encontrado las diferentes funciones del sistema bien integradas
- He encontrado las diferentes funciones del sistema bien integradas
- Creo que había inconsistencias en el sistema
- Creo que la mayoría de la gente aprendería rápidamente a usar el sistema
- He encontrado el sistema muy incómodo al usarlo
- Me sentí muy agusto utilizando el sistema
- Necesito aprender un montón de cosas antes de utilizar el sistema

Cada usuario debe responder con un grado de acuerdo de 1 a 5 y como se ve se alternan las cuestiones negativas y positivas. Al terminar se suman los puntos (los negativos a la inversa obviamente) y si se obtiene un percentil que según diversas investigaciones debe ser superior a 68 para considerar el sistema usable.

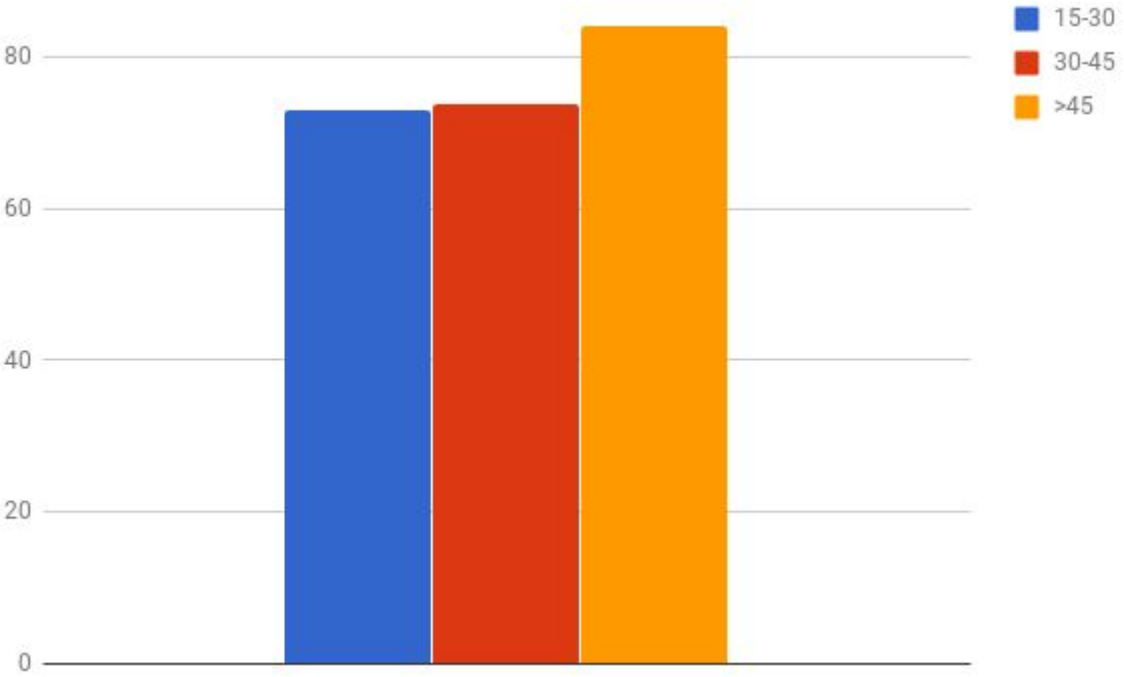
Resultados del test

La puntuación SUS de la interfaz ha sido de 76.25 puntos por tanto es posible asegurar que el sistema es usable. Ha habido 25 usuarios que han probado la herramienta durante unos 10 minutos y después respondieron al test y a un pequeño cuestionario personal (edad y sexo). A continuación el desglose de las respuestas por pregunta y tipo de usuario.



Desglose de respuestas por pregunta

Por sexo los hombres otorgan un SUS medio de 76.2 y las mujeres de 76.3 por lo que podemos decir que no hay diferencias apreciables entre sexos.



Desglose por edades

Por edades parece que a los mayores de 45 años les resulta más usable la herramienta, a futuro se podrían realizar más análisis para tratar de descubrir por qué y averiguar si son el mejor público para esta herramienta.

Los servicios

En el marco de este trabajo no se han realizado tests sobre los servicios, ha sido la propia empresa en la que se realizaron (TweetBinder) la encargada de analizarlos, revisarlos y testearlos. Sin embargo cabe recalcar que actualmente estos servicios son utilizados por la empresa en producción y los clientes pueden realizar informes históricos de Twitter sin límite de tiempo y con un coste económico razonable.

Conclusión

Viendo estos resultados se puede afirmar que los objetivos del trabajo se han cumplido completamente.
Recordamos que los objetivos eran:

- Desarrollar una herramienta que permita recoger información de Twitter desde el comienzo de la red social de forma automática
- Conseguir un coste que sea mínimo o incluso gratuito.
- Desarrollar y testear una interfaz para controlar la herramienta

Se ha conseguido desarrollar una herramienta automática que recupera tweets sin importar su fecha de publicación a coste cero, como mucho el que conlleva hacer el informa que no son más de 3 o 4 clicks. También se ha diseñado, implementado y testeado una interfaz para utilizarla que ha pasado el test de usabilidad con éxito.

Conclusiones del desarrollo

La solución final alcanzada ha sido muy satisfactoria, somos capaces de recuperar tweets emitidos desde el comienzo de Twitter (año 2006) y de hacerlo de una forma rápida, automática y escalable.

Este trabajo está siendo monetizado por la empresa Tweet Binder con gran éxito, cada día clientes por todo el mundo compran informes históricos realizados con esta herramienta por el precio de 220€ para conocer mejor que se dice sobre ellos, medir resultados de sus campañas y tener referencias para comparar informes más modernos.

Personalmente mi crecimiento durante la realización de este trabajo ha sido muy grande, he aprendido muchísimo sobre como funcionan las redes sociales, las diferencias que hay entre ellas y cómo casi todas se explican por los diferentes modelos de negocio que existen.

A nivel técnico manejar tantas tecnologías como el entorno de Node junto a Npm, la persistencia en una base de datos no relacional como MongoDB, el paso de mensajes con colas mediante SQS, la gestión de los entornos con Docker o el despliegue de los servicios mediante contenedores en Heroku son cosas que a buen seguro me serán útiles en el futuro y que considero que me hacen un desarrollador más completo.
Más allá de tecnologías en este proyecto he aprendido a gestionar un proyecto informático real, a desarrollar en el marco de una empresa donde todo es urgente y en ocasiones no hay que buscar la mejor solución sino la más rápida y también como plantear un proyecto desde la identificación del problema hasta el despliegue y testeo de la solución propuesta.

En conclusión, creo que he sido capaz de desarrollar una herramienta realmente interesante, novedosa y competitiva en el mundo real y que durante el proceso he adquirido numerosas competencias -tanto técnicas como personales- que me serán útiles como egresado.

Líneas futuras

Pensando en como se podría ampliar este proyecto cabe pensar en varias alternativas, por un lado Twitter está inmerso en el lanzamiento de una nueva API para empresas, no tan cara como la actual y que se supone va a permitir la realización de informes históricos así que podríamos analizarla y ver si su coste es razonable.

Por otra parte la interfaz de la aplicación es bastante básica y es posible aumentar sus funcionalidades u ofrecer más estadísticas de cada informe.

Por otro lado resulta especialmente interesante ampliar la plataforma al análisis de otras redes sociales como Instagram (recientemente restringió el acceso a su API y ahora ofrece

más datos al visitarla por internet (de manera similar a lo que se hace en este trabajo con Twitter) que mediante su API.

Bibliografía

[1] M. MASSE. “Chapter 1. Introduction”, en *REST API Design Rulebook*. Ed. O'Reilly Media, Inc. 2011, pp 4-5.

[2] The JavaScript Object Notation (JSON) Data Interchange Format, RFC 8259, 2017 [En línea] Disponible en https://tools.ietf.org/html/rfc8259.[Accedido: 20-may-2018]

[3] “Api reference Index -- Twitter Developers”, 2018. [En línea]. Disponible en https://developer.twitter.com/en/docs/api-reference-index. [Accedido 20-may-2018]

[4] The OAUTH 1.0 protocol, RFC 5849, 2010 [En línea] Disponible en https://tools.ietf.org/html/rfc5849.[Accedido: 20-may-2018]

[5] “Cómo utilizar la búsqueda avanzada”,2018. [En línea]. Disponible en https://help.twitter.com/es/using-twitter/twitter-advanced-search. [Accedido 20-may-2018]

[6] “About | Nodejs”, 2018 [En línea]. Disponible en https://nodejs.org/en/about/. [Accedido 20-may-2018]

[7] “Authorization Guide | Spotify Developers”, 2018, [En línea]. Disponible en https://developer.spotify.com/documentation/general/guides/authorization-guide/. [Accedido 6-jun-2018]

[8] Scripting Media Types, RFC 4329,2006 [En línea]. Disponible en https://tools.ietf.org/html/rfc4329.[Accedido: 20-may-2018]

[9] “Bootstrap”, 2018 [En línea]. Disponible en https://getbootstrap.com/. [Accedido 20-may-2018]

[10] The OAUTH 1.0 protocol, RFC 7797, 2016 [En línea] Disponible en https://tools.ietf.org/html/rfc7797. [Accedido: 20-may-2018]

[11] J.R. Lewis (2018) The System Usability Scale: Past, Present, and Future, International Journal of Human–Computer Interaction, DOI: 10.1080/10447318.2018.1455307