TweetManager

Grupo *Woody*:

Víctor Lafaja Sanz (680794)
Santiago Morón Borraz (681417)
Iván Escuín González (684146)
Jorge Aguarón de Blas (698895)

29 de Agosto de 2018

ÍNDICE

1. Resumen del proyecto

Este proyecto tiene como objetivo desarrollar un sistema que permite gestionar, mediante una interfaz típica de control de acceso de usuarios, una o varias cuentas de *Twitter*[1].

Para ello, se ha dividido el proyecto en *cuatro partes*:

1. *Sistema de control de acceso*: se restringirá o permitirá el acceso de un usuario a la aplicación mediante la autenticación por cuenta propia de la aplicación o mediante el acceso con cuentas de *Gmail, Facebook o Twitter*.
2. *Aplicación de usuario*: al acceder un usuario a la aplicación tendrá una vista principal del panel de control desde el que podrán gestionar sus cuentas de Twitter (añadir, eliminar).
3. *Análisis y estadísticas*: el usuario tendrá acceso a una vista desde la que tendrá una visión completa de cómo van sus cuentas.
4. *Interfaz de administración*: la aplicación tendrá un usuario administrador que, mediante su interfaz, se permitirá la gestión y administración de usuarios.

A nivel de implementación, se ha utilizado el stack MEAN para programar el sistema por completo.

[1] Es una red social en la que sus usuarios publican e interactúan mediante mensajes conocidos como *"tweets"*.

2. Propuestas similares

En el mundo competitivo de hoy en donde las empresas lanzan nuevos productos al mercado cada vez con mayor frecuencia, y en donde cada vez aparecen más empresas competidoras, se hace prácticamente una obligación realizar el análisis de la competencia.

La competencia puede ser directa cuando se trata de empresas que producen o venden productos similares a los de uno, o indirecta cuando se trata de empresas que producen o venden productos sustitutos a los de uno. Pero por lo general, al menos en el caso de las pequeñas y medianas empresas, para realizar el análisis de la competencia solo se suele tomar en cuenta la competencia directa.

En este apartado se van a analizar diferentes aplicaciones que hay en el mercado que tengan una funcionalidad similar (competencia directa) a *TweetManager*:

2.1. Hootsuite

Una de las herramientas *social media* más popular. Permite dirigir diferentes perfiles de redes sociales, programas mensajes o estados, publicar mensajes en varias redes de forma simultánea, etc. Además, su interfaz permite una cómoda visualización de cada plataforma. Una de las principales ventajas de *HootSuite* y, el objeto por el que fue creado, es la de la programación de publicaciones en el tiempo.

2.2. TweetDeck

Muchos lo consideran la competencia de *Hootsuite*, ya que las funciones de ambas herramientas social media son similares. Sin embargo, esta tiene una gran flexibilidad para su uso con todo tipo de plataformas y conecta a todos tus contactos a través de las diferentes redes sociales.

2.3. SpreadFast

SpreadFast es un sistema de manejo de redes sociales que te ayuda a organizar exitosamente tu presencia en las redes sociales y a poner en acción tus planes de una buena estrategia online. Este software activa un monitoreo y una aplicación social, haciendo posible organizar tu trabajo en un calendario global y utilizar herramientas organizacionales. Además, podrás utilizar contenido compartido y análisis comparativos para rastrear tanto tu trabajo como tus progresos. El contar con acceso móvil es otra prueba de la flexibilidad que esta plataforma ofrece.

2.4. Bit.ly

Una herramienta muy útil para ahorrar espacio y dar mejor estética a las publicaciones. Permite acortar las URL y, además, controlar las estadísticas de estos links. Estas funciones pueden mejorar la armonía visual, evitando enlaces demasiado largos, en redes como Facebook, pero en Twitter, se convierte en algo esencial para poder realizar publicaciones, ya que la extensión de contenidos se ve limitada a 140 caracteres.

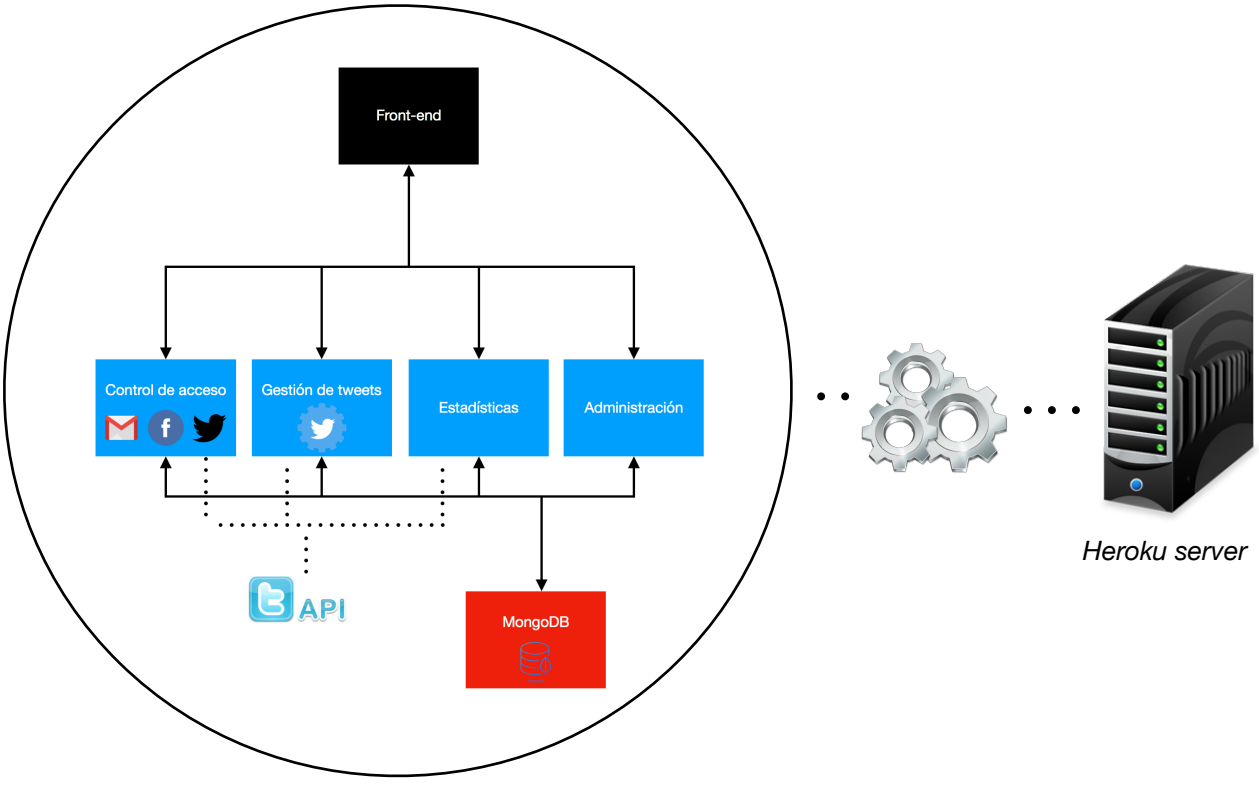
Hay muchas más aplicaciones que podríamos considerar competencia como: *Twiterfeed, Buffer, Blurtster, Sysomos…*

Puesto que que la competencia en la actualidad es muy grande, el objetivo debe ser implementar alguna funcionalidad nueva o tratar de hacer una aplicación más usable y accesible que las que hay en el mercado.

3. Arquitectura de alto nivel

A continuación se van a explicar los puntos más importantes de la arquitectura de la aplicación:

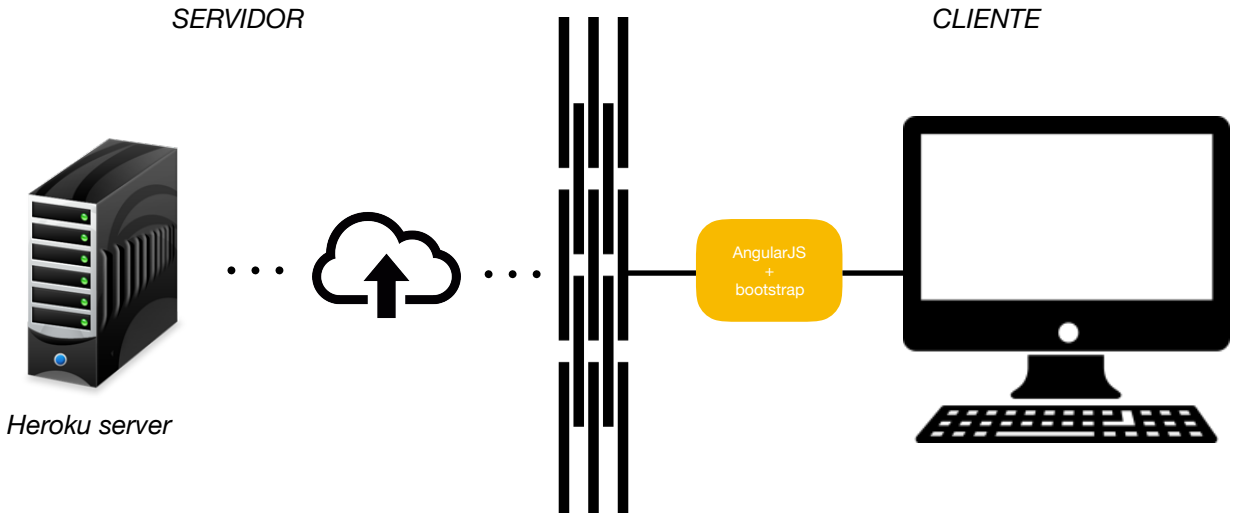
- El servicio principal de la aplicación es el servicio de gestión de tweets.
- Existen otros tres servicios adicionales, que se complementan entre ellos, como son: el control de acceso, análisis y estadísticas y administración.
- Todos estos servicios hacen uso de una base de datos no relacional[2] como es *MongoDB*.
- El control de acceso además de implementar una autenticación propia también hace uso de los sistemas de autenticación de Facebook, Google y Twitter.
- El sistema de autenticación de Twitter, el servicio de gestión de tweets y el servicio de estadísticas hacen uso de la *API de Twitter*.
- La parte de *front-end* se encarga de comunicarse con cada uno de los servicios y hacer las peticiones necesarias según las acciones solicitadas por el usuario.



Todos estos servicios se ejecutan en *Heroku*, uno de los *PaaS (Platform as Service)* más utilizados en la actualidad en entornos empresariales por su fuerte enfoque en resolver el despliegue de una aplicación. Además, Heroku permite manejar los servidores y sus

[2] Las bases de datos NoSQL están diseñadas específicamente para modelos de datos específicos y tienen esquemas flexibles para crear aplicaciones modernas. Las bases de datos NoSQL son ampliamente reconocidas porque son fáciles de desarrollar, su funcionalidad y el desempeño a escala. Usan una variedad de modelos de datos, que incluyen documentos, gráficos, clave-valor, en-memoria y búsqueda.

configuraciones, escalamiento y la administración. A Heroku basta con decirle qué lenguaje de backend se está utilizando o qué base de datos se va a utilizar y ya sólo hay que preocuparse únicamente por el desarrollo de la aplicación.



Los datos llegan al cliente a través de la conexión con los controladores. Cada una de las pantallas que muestran datos por pantalla utiliza un controlador que es el que se encarga de ejecutar una función en el servicio al que se conecta, que a su vez envía una petición a nuestra API (al servidor), el cual en muchos casos va a mandar una petición a la API de *Twitter*. Estos datos se devuelven a una variable del controlador asociado a la vista, que es accesible desde la vista.

4. Detalles de los componentes arquitecturales

4.1. Control de acceso

Mediante este componente, se permitirá o no el acceso de un usuario a la aplicación. La aplicación permite el registro de un usuario, para lo que solicitará el nombre, apellido y email del usuario. A continuación, el sistema enviará al correo electrónico especificado en el registro una contraseña generada de forma pseudoaleatoria para el primer acceso del usuario. Una vez el usuario inicia sesión por primera vez, se le obliga a cambiar la contraseña.

También está permitido acceder a la aplicación utilizando una cuenta de Gmail o Twitter.

4.2. Gestión de tweets

Este componente trabaja continuamente con el API oficial REST de Twitter. Se encarga de realizar las peticiones *GET y POST*[3], además de la interacción con el *Streaming API* de Twitter.

4.3. Estadísticas

Mediante este servicio se le ofrece al usuario poder ver de forma gráfica:

- De los últimos 300 tweets de cada cuenta asociada, el número de tweets publicados cada día del mes entre todas sus cuentas.
- De los últimos 300 tweets de cada cuenta asociada, el número de tweets publicados cada día de la semana entre todas sus cuentas.

4.4. Administración

Existe un usuario *administrador* de la aplicación que permite la gestión y administración de usuarios desde su interfaz propia.

4.5. Base de datos

Es MongoDB. En ella se almacena toda la información de la aplicación: usuarios, estadísticas, tweets programados y URLs acortadas.

4.6. Front-end (AngularJS + bootstrap)

Esta parte de la aplicación es la encargada de presentar toda la información al usuario. El usuario interacciona con esta parte de la aplicación, la cual realiza las llamadas al resto de componentes para obtener los datos pedidos por el usuario. Estos datos son tratados

[3] *https://www.w3schools.com/tags/ref_httpmethods.asp*

para presentárselos al usuario de manera que el uso de la aplicación sea sencillo y agradable. *AngularJS* se encarga de la interacción entre la interfaz y el servidor; y *bootstrap* sirve para renderizar la página web y es un componente estético.

5. Modelo de datos

El modelo de datos del proyecto consta de *cuatro* tablas:

5.1. Usuario

Almacena los usuarios registrados en la aplicación, así como las cuentas de *Twitter* asociados a ellos.

Atributo	Tipo de dato	Descripción
nombre	String	Nombre de pila del usuario
apellidos	String	Apellidos del usuario
email	String	Email del usuario. No puede existir dos iguales.
password	String	Contraseña del usuario encriptada para evitar su lectura
admin	Boolean	Indica si el usuario tiene permisos de *administrador*
cuentas	Map	Almacena los datos relevantes de las cuentas de Twitter
cuentas.public_name	String	Nombre público de la cuenta de Twitter
cuentas.email	String	Email asociado a la cuenta de Twitter
cuentas.account_name	String	Nombre de usuario de la cuenta de Twitter
cuentas.hashtags	String array	Etiquetas seguidas por la cuenta de Twitter
cuentas.token	String	Token del usuario para la comunicación con la API de Twitter
cuentas.tokenSecret	String	Token secreto del usuario para la comunicación con la API de Twitter
origen	String array	Opciones con las cuales ha iniciado sesión el usuario en la aplicación
primerAcceso	Boolean	Indica si ha iniciado sesión por primera vez en la aplicación
entradaApp	Date	Fecha de registro en la aplicación
ultimoAcceso	Date	Fecha del último acceso a la aplicación

5.2. Estadísticas

Recoge estadísticas asociadas al usuario respecto a su número de accesos, su estado actual… Existe un objeto <estadísticas> asociado a cada usuario por su id de cuenta.

Atributo	Tipo de dato	Descripción
idCuenta	String	Identificador del usuario asociado a la estadística
email	String	Email del usuario asociado a la cuenta
active	Boolean	Indica si el usuario se ha dado de baja o sigue activo
numTweets	Number	Número de tweets publicados por el usuario entre todas sus cuentas
tweets	JSON array	Vector de estadísticas referentes a los tweets del usuario
tweets.account_name	String	Nombre de usuario de la cuenta de Twitter del tweet publicado
tweets.time	Date	Fecha de creación del tweet
tweets.tweetLength	Number	Número de caracteres del tweet
fechaBaja	Date	Fecha de eliminación del usuario de la aplicación
fechaAlta	Date	Fecha en la que el usuario se registra en la aplicación
numAccess	Number	Número de accesos del usuario a la aplicación

5.3. URLs acortadas

Almacena la relación entre una URL acortada y la original.

Atributo	Tipo de dato	Descripción
originalUrl	String	URL completa del sitio web
urlCode	String	Código aleatorio generado que identifica el objeto
shortUrl	String	URL acortada resultante
createdAt	Date	Fecha de creación de la URL
updatedAt	Date	Fecha de última actualización de la URL

5.4. Tweets programados

Almacena los tweets programados por el usuario.

Atributo	Tipo de dato	Descripción
usuario	String	Email de la cuenta de Twitter que ha programado el tweet
cuenta	String	Nombre de la cuenta de Twitter que ha programado el tweet
creado	Date	Fecha en la que ha sido creado el tweet
trigger	Date	Fecha en la que se publicará el tweet

Atributo	Tipo de dato	Descripción
status	String	Indica si está enviado o no
text	String	Texto asociado al tweet

6. API REST

6.1. users.js

Petición	Descripción
GET /	Devuelve el usuario pasado como parámetro
GET /users	Devuelve todos los usuarios de la aplicación
GET /deleteUser	Elimina el usuario pasado como parámetro
POST /	Crea un usuario con los parámetros recibidos
POST /recover	Trata una petición para recuperar la contraseña
PUT /	Asignación de la nueva contraseña al iniciar sesión por primera vez
PUT /users	Actualiza un usuario con los parámetros recibidos

6.2. index.js

Petición	Descripción
GET /*	Redirige al usuario dependiendo de si está logueado o no
GET /login	Inicia sesión en la aplicación por el método local
GET /loginGoogle	Inicia sesión en la aplicación con cuenta de Google
GET /loginGoogle/callback	Redirige al usuario a la aplicación tras iniciar sesión con cuenta de Google
GET /loginFacebook	Inicia sesión en la aplicación con cuenta de Facebook
GET /loginFacebook/callback	Redirige al usuario a la aplicación tras iniciar sesión con cuenta de Facebook
GET /loginTwitter	Inicia sesión en la aplicación con cuenta de Twitter
GET /loginTwitter/callback	Redirige al usuario a la aplicación tras iniciar sesión con cuenta de Twitter
GET /session	Devuelve el email del usuario logueado
GET /logout	Elimina cookies y cierra sesión

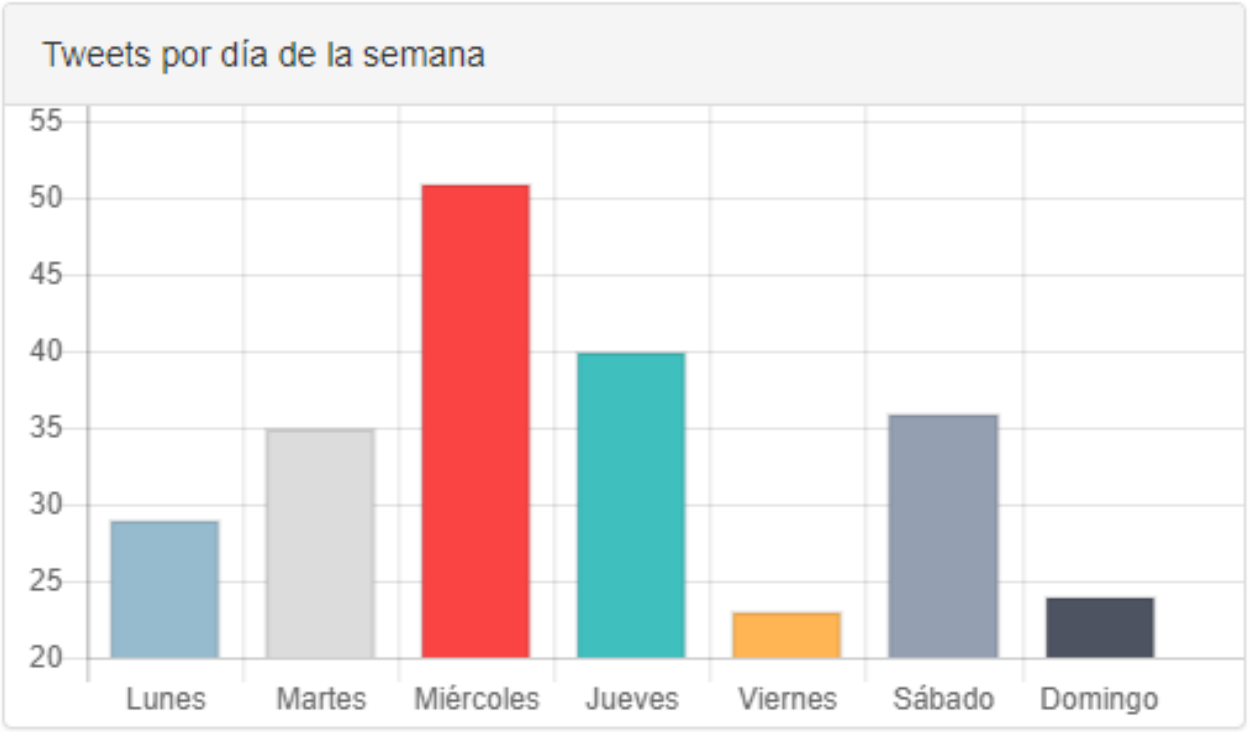
6.3. account.js

Petición	Descripción
GET /	Devuelve un array con la información pública de las cuentas de Twitter del usuario
GET /accTokens	Añade una cuenta de Twitter al usuario

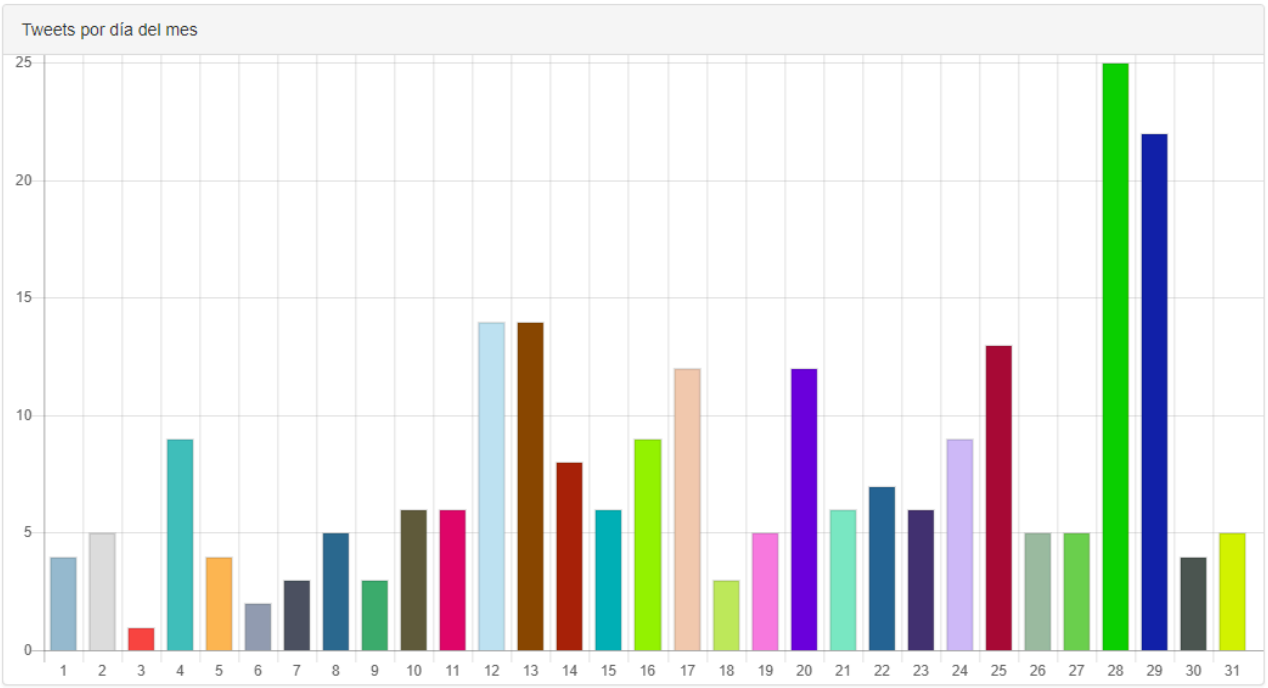
Petición	Descripción
GET /tokens/callback	Redirige a la aplicación tras añadir una cuenta de Twitter
GET /twits/home/:account/:user	Devuelve los tweets correspondientes al home de la cuenta de Twitter
GET /twits/home/:user	Devuelve los tweets publicados por el usuario desde todas sus cuentas
GET /twits/user/:account/:user	Recibe los últimos 20 tweets publicados por el usuario
GET /twits/mentions/:account/:user	Recibe las 20 últimas menciones del usuario
GET /twits/programados/:account/:user	Devuelve los tweets programados de la cuenta
GET /twits/retweets/:account/:user	Recibe los 20 últimos retweets de la cuenta
GET /twits/retweets/:account/:user	Recibe los 50 últimos tweets marcados como favoritos de la cuenta
GET /twits/hashtags/:account/:user	Recibe los hashtags asociados a la cuenta del usuario
POST /twits/newTweet/:account/:user	Publica un tweet
POST /addUrl	Guarda la URL acortada
POST /twits/programados/:account/:user	Programa un tweet
POST /twits/newProgTweet/:account/:user	Publica un tweet programado y actualiza estadísticas
POST /twits/hashtags/:account/:user	Agrega a la cuenta un hashtag
DELETE /twits/hashtags/:account/:user	Elimina un hashtag de la cuenta de usuario
PUT /insertAcc	Añade una cuenta de Twitter al usuario
PUT /deleteAcc	Elimina una cuenta de Twitter del usuario

7. Analíticas

- *De los últimos 300 tweets de cada cuenta asociada, el número de tweets publicados cada día de la semana entre todas sus cuentas. Esto permite al usuario conocer el día de la semana que más activo está en Twitter. En el ejemplo siguiente, sería el miércoles.*



- *De los últimos 300 tweets de cada cuenta asociada, el número de tweets publicados cada día del mes entre todas sus cuentas. Esto permite al usuario conocer el día del mes que más activo está en Twitter. En el ejemplo siguiente, sería el día 28.*



8. Implementación

La aplicación se ha desarrollado con la arquitectura del *stack MEAN*[4] siguiendo el patrón *model-view-controller*[5]. El servidor ha sido implementado para el entorno de ejecución *Node.js*. La ejecución se realizaba localmente hasta que se ha configurado una instancia en *Heroku* para la entrega.

Se ha utilizado *express* para el manejo de las rutas de la aplicación. Para ello se han definido distintos ficheros de rutas que se encargan de cablear las rutas de la aplicación con su controlador específico.

Para el control de accesos y añadir cuentas asociadas al usuario, se ha utilizado passports. Por otra parte, un usuario puede registrarse con una dirección de correo eléctrónico, pero necesitara aportar información adicional (nombre y apellidos) y rellenar un captcha. Esto se ha hecho así para evitar la creación de cuentas por parte de *bots o spammers*. Tras registrarse un usuario mediante el método local, se le envía al correo introducido una contraseña generada de forma pseudoaleatoria que se le obligará a modificar en el primer inicio de sesión.

Tras el inicio de sesión el usuario llega a una página en la que puede gestionar sus cuentas de Twiiter asociadas (añadir, eliminar) y desde la cual puede acceder a la vista de los tweets de una cuenta o al análisis de las estadísticas de todas sus cuentas asociadas. El usuario tiene permitido modificar su contraseña desde el perfil y darse de baja.

Por otro lado, se creó un usuario *administrador* que tiene acceso a una interfaz única para él desde la cuál puede gestionar y administrar usuarios. Esta interfaz es un *SPA (Single Page Application)* debido a la baja tasa de actualización de los datos asociados a las vistas.

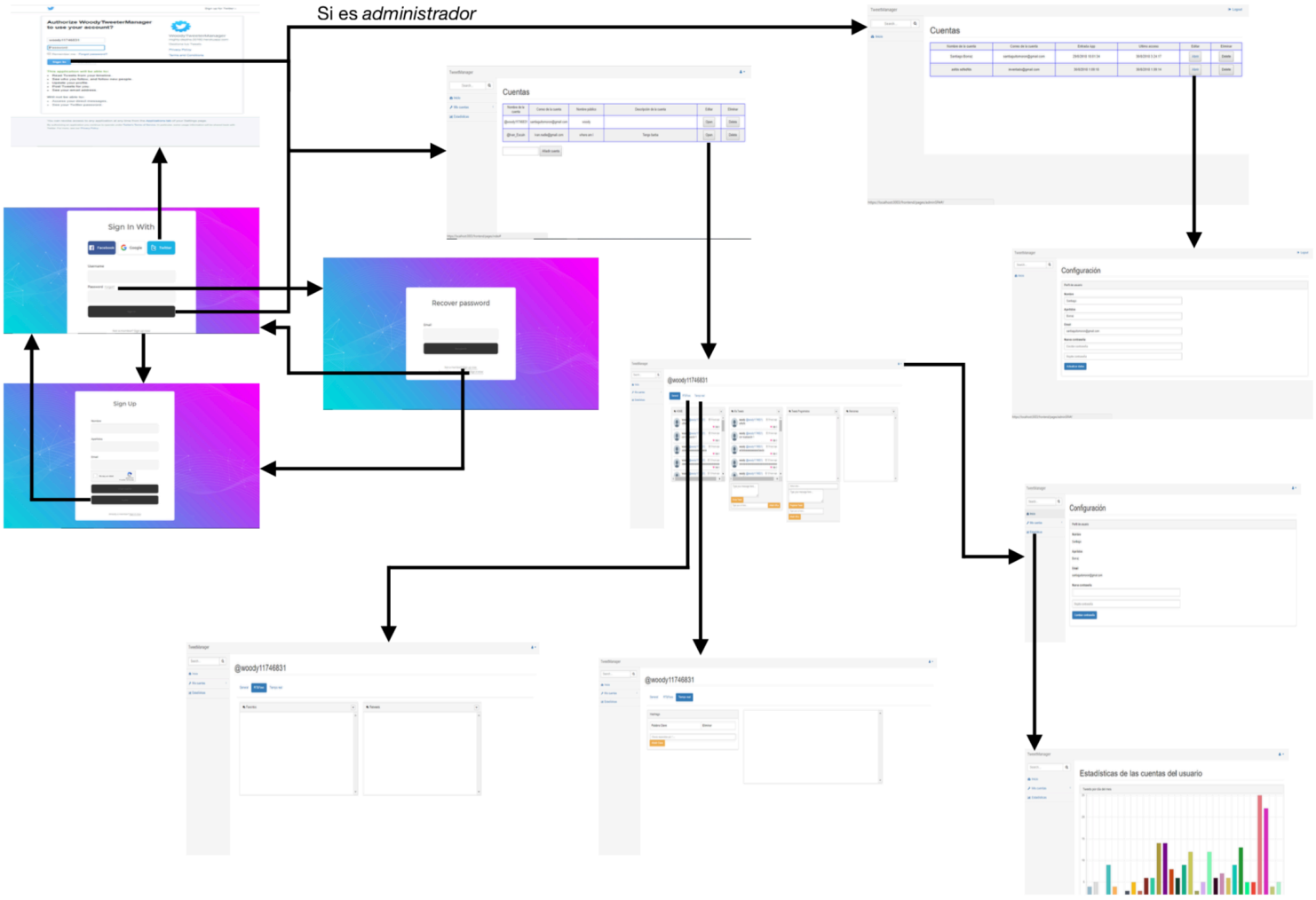
Se ha utilizado *bootstrap* para crear una plantilla agradable e intuitiva de la aplicación y con diseño *responsive*. Mientras, se ha utilizado *AngularJS* para hacer las página dinámicas y extraer los datos de la aplicación para su posterior muestra al usuario. Cada *HTML* que incluía *AngularJS* tiene asociado uno o más controladores de *AngularJS* para el tratamiento de los datos y la realización de peticiones.

---

[4] MongoDB, Express, Angular y Node.js

5 Es un patrón de arquitectura de software, que separa los datos y la lógica de negocio de una aplicación de su representación y el módulo encargado de gestionar los eventos y las comunicaciones.

9. Mapa de navegación



10. Despliegue del sistema

Para el despliegue local de la aplicación se deben realizar los siguientes pasos:

1. Clonar el repositorio de GitHub de la aplicación y situarnos en él:

```
C:\Users\Santiago\Desktop\despliegue>git clone https://github.com/crzyivo/TweetManagerWoody.git
Cloning into 'TweetManagerWoody'...
Username for 'https://github.com': insige
Password for 'https://insige@github.com':
remote: Counting objects: 4778, done.
remote: Compressing objects: 100% (609/609), done.
remote: Total 4778 (delta 432), reused 750 (delta 282), pack-reused 3854
Receiving objects: 100% (4778/4778), 9.49 MiB | 2.19 MiB/s, done.
Resolving deltas: 100% (1947/1947), done.
Checking connectivity... done.

C:\Users\Santiago\Desktop\despliegue>cd TweetManagerWoody
```

2. Cambiar a la rama de la base de datos, llamada "*mongo*":

```
C:\Users\Santiago\Desktop\despliegue\TweetManagerWoody>git checkout mongo
Branch mongo set up to track remote branch mongo from origin.
Switched to a new branch 'mongo'
```

3. Instalar los paquetes necesarios mediante *Node.js*:

```
C:\Users\Santiago\Desktop\despliegue\TweetManagerWoody>npm install
```

4. Salir del directorio donde se encuentra la base de datos y cambiar el nombre del directorio actual por otro (*mongo*, por ejemplo).
5. Volver a clonar el repositorio (pero esta vez se mantendrá la rama *master* como seleccionada).
6. Entrar en el directorio del repositorio clonado e instalar los paquetes necesarios para la ejecución de la aplicación:

```
C:\Users\Santiago\Desktop\despliegue\TweetManagerWoody>npm install
```

7. Acceder al directorio *public/frontend* e instalar los módulos correspondientes a las vistas internas de la aplicación:

```
C:\Users\Santiago\Desktop\despliegue\TweetManagerWoody>cd public

C:\Users\Santiago\Desktop\despliegue\TweetManagerWoody\public>cd frontend

C:\Users\Santiago\Desktop\despliegue\TweetManagerWoody\public\frontend>npm install
npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@^1.0.0 (node_modules\chokidar\node_modules\fsevents):
npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.4: wanted {"os":"darwin","arch":"any"} (cur
rent: {"os":"win32","arch":"x64"})
npm WARN startbootstrap-sb-admin-2@3.3.7 license should be a valid SPDX license expression
```

8. Iniciar la aplicación:

```
C:\Users\Santiago\Desktop\despliegue\TweetManagerWoody>npm start
```

9. Acceder al directorio raíz del proyecto en el que está la base de datos y ejecutar también el comando *npm start*.
10. De esta forma, ya estaría desplegada la aplicación y únicamente se precisaría viajar desde el navegador a la dirección *http:///localhost:3003*

11. Validación

El *testing* de la aplicación se ha realizado de forma manual. Se han probado las diferentes funcionalidades de la aplicación de forma repetida y con diferentes parámetros para validar el correcto funcionamiento del sistema. Algunas de los tests más importantes son:

- ✓ *Creación correcta de usuarios*: se introducen correctamente los campos del registro vía local y el resultado ha sido exitoso.
- ✓ *Creación incorrecta de usuario*: se ha intentado introducir un email de formato incorrecto en el campo de registro de usuario. La validación del input muestra un formato de error debido al formato incorrecto del campo del email, incluyendo una explicación de como rellenarlo correctamente.
- ✓ *Añadir correo de usuario ya existente*: se ha intentado introducir un email de un usuario ya existente. La validación del correo muestra un mensaje de error indicando que el email ya está en uso.
- ✓ *Añadir una cuenta de Twitter*: se ha intentado añadir una cuenta de Twitter con las correctas credenciales y el resultado ha sido exitoso.
- ✓ *Añadir una cuenta de Twitter con credenciales erróneas*: se ha intentado añadir una cuenta de Twitter con las credenciales incorrectas y la API de Twitter ha comprobado que están son erróneas y ha recargado la página de introducción de credenciales para volver a poder introducir las credenciales correctas.
- ✓ *Modificar desde el perfil de administrador un usuario*: se ha intentado modificar los datos referentes a una cuenta de usuario y el resultado ha sido exitoso.
- ✓ *Modificar desde el perfil de administrador el email de un usuario por otro email ya existente*: se ha intentado cambiar el email de un usuario por el de un email ya existente y el sistema ha denegado el cambio y ha mostrado un mensaje por pantalla en el que avisaba del fallo en la modificación.

12. Análisis de problemas

En los dos apartados posteriores se va a proceder a documentar los diferentes problemas encontrados:

12.1. Problemas encontrados durante el desarrollo

Para la autenticación con *Twitter* se utilizaron diferentes herramientas y llamadas de la *API REST de Twitter.* Tras varios intentos fallidos se consiguió realizar la autenticación por medio de *Passport*. Una vez conseguido autenticar al usuario para entrar en la aplicación vía cuenta de twitter se pensó en utilizar el mismo método para introducir las cuentas de twitter definiendo para ello una nueva estrategia de *Passport*. El problema que derivó de definir una nueva estrategia fue el conflicto entre la estrategia antigua y la nueva que daba error y no permitía añadir cuentas. Esto se solucionó encapsulando la creación de la nueva estrategia dentro de un método de forma que no sufriera conflicto al redirigir al usuario desde index.

Para el inicio de sesión con Passport se encontró el problema de la imposibilidad de definir un controlador de angular que gestionase la redirección una vez logeado con Google, Twitter o Facebook. Esto se debía a que se daba un problema de Control de Acceso y no se nos permitía redirigir a la aplicación desde el controlador de angular de index. En consecuencia, el usuario que se creaba mediante el login normal con su correo asignado tenía asociado un contexto (un correo electrónico) que los usuarios logeados por APIs externas no tenían.

Mediante una llamada a GET /session se soluciono el problema, devolviendo el contexto situado en la variable req.user creada tras iniciar sesión con passport.

12.2. Análisis de problemas potenciales

En cuanto al tema de estadísticas se refiere, el API oficial REST de Twitter tiene distintas limitaciones, todas ellas relacionadas con el número de peticiones que se permiten hacer o el número de datos que muestra.

El API de twitter, permite realizar como máximo 15 peticiones GET cada 15 minutos. Esto es una clara limitación que puede encontrar nuestro sistema provocando que si se interactúa rápidamente con la aplicación superando el número de peticiones, no se muestren los datos.

Por otro lado, otra clara limitación que el API de Twitter nos provoca es el número de resultados devueltos por las peticiones. En el caso de que un usuario publique muchos tweets no se podrá mostrar información de todos ellos, por lo que habrá limitaciones a la hora de hacer las estadísticas y gráficas.

13. Distribución del tiempo

Integrante del grupo	Documentación de las tecnologías	Comprensión de la aplicación	Implementación	Pruebas	Memoria	Horas
Víctor	10	7	40	5	8	70
Santiago	20	5	60	8	8	101
Iván	15	10	60	8	4	97
Jorge	10	5	40	4	4	63
HORAS TOTALES	**55**	**27**	**200**	**25**	**24**	**331**

14. Conclusiones

La aplicación desarrollada permite gestionar, mediante una interfaz típica de control de acceso de usuarios, una o varias cuentas de Twitter, que era el objetivo principal de este proyecto.

A nivel de implementación, hemos empleado la tecnología stock MEAN para programar el sistema por completo, que era exactamente lo que se nos pedía en el enunciado. Esta tecnología ha hecho uso de la API de Twitter y hemos sido capaces de gestionar/ administrar varias cuentas de Twitter simultáneamente.

14.1. Conclusiones del proyecto

Gracias a las clases de prácticas, el empleo de la tecnología MEAN ha resultado más sencillo puesto que en ellas se explicaban los conceptos de ésta. De esta forma, todos los miembros del grupo partíamos de una misma base, ya que ninguno teníamos conocimiento alguno de esta tecnología.

Debido a una mala organización durante el segundo cuatrimestre del curso, la aplicación no pudo estar lista para su presentación en la primera convocatorio, por lo que la hemos tenido que posponer para la segunda convocatoria.

El sistema cumple con la mayoría de los requisitos de la aplicación.

El proyecto ha servido para profundizar en el desarrollo de aplicaciones web y adquirir conocimientos sobre el panorama actual.

15. Valoraciones

15.1. Valoración personal del grupo

Víctor

El grupo con el que he desarrollado el proyecto me ha parecido verdaderamente potente dados sus conocimientos, su rápida respuesta a los problemas que se plantean y el empeño, trabajo y sacrificio que demuestran en los momentos difíciles creando un ambiente de trabajo agradable y de apoyo mutuo.

Me gustaría destacar la labor realizada en los últimos días por Santiago e Iván, quienes han cogido las riendas del equipo y han sabido tirar de él en los instantes finales. Especialmente dar las gracias a Santiago por el esfuerzo que hizo para poder quedar conmigo algunos días para que me explicara más en detalle las tecnologías que él había estado estudiando durante el verano en Hiberus. De esta forma, podíamos trabajar los dos conjuntamente desde su ordenador siguiendo la metodología ágil de desarrollo de software que se conoce como *Pair Programming*.

Santiago

Personalmente me ha parecido una buena experiencia, explorar desde dentro el funcionamiento de un equipo de desarrollo desde distintos puntos de vista. También sirve para probar el trabajo en equipo "real" y los problemas o tensiones que pueden surgir entre miembros del equipo. Afortunadamente, debido a la positividad y actitud del equipo en todo momento, no ha habido ningún problema serio más allá de alguna bronca amistosa ocasional.

Iván

Tras un comienzo tardío del proyecto en la primera convocatoria, sinceramente me temía lo peor debido a que si ya nos resultaba difícil cuadrar horarios durante el curso, en verano iba a ser muchísimo peor puesto que unos nos íbamos de vacaciones, otros al pueblo, otros comenzaban prácticas. Sin embargo, he de reconocer que la respuesta del grupo en estas dos últimas semanas me convenció y hemos conseguido terminar la mayoría de los puntos pedidos en el proyecto.

Jorge

Tras un inicio del proyecto un tanto regular, el grupo supo sobreponerse y a pesar de las dificultades que teníamos para cuadrar horarios, pudimos quedar en algunas ocasiones para poner ideas en común y ayudarnos entre nosotros. El grupo, en general, fue de menos a más.

15.2. Valoración personal de cada miembro

Víctor

El enfoque de la asignatura me parece muy bueno, ya que implementar una aplicación web, puede servirnos de mucha utilidad en un futuro en un posible trabajo.

Este proyecto me ha parecido bastante más pesado que otros de diferentes asignaturas de la titulación ya que al tener un desconocimiento total de las tecnologías empleadas resultaba mucho más difícil avanzar.

Sin embargo, he de decir que he disfrutado mucho en el transcurso de la asignatura, tanto en el desarrollo en equipo del proyecto como en las clases y las prácticas realizadas. Llegado el final estoy encantado de la experiencia adquirida ya que es la muy cercana a lo que vamos a encontrarnos en el mundo laboral cuando trabajemos en grupos de gente. Personalmente considero que la asignatura enseña algunos aspectos del mundo laboral que no se viven en el resto de asignaturas.

Santiago

Considero el proyecto realizado una experiencia previa al entorno laboral muy instructiva dado que nos permite observar más de cerca cómo se desenvuelven en su día a día los grupos de desarrolladores y nos permite tener una primera toma de contacto con herramientas verdaderamente útiles para nuestro futuro.

Casualmente, empecé prácticas este verano en Zaragoza en la empresa Hiberus y estuvieron durante las primeras semanas enseñándome acerca de la tecnología stack MEAN, lo que hizo que mi tiempo de documentación fuera mucho mayor que el de mis compañeros.

Iván

Considero muy positivamente la realización de este trabajo ya que nunca había trabajado con esta tecnología y actualmente se utiliza en numerosos proyectos.

Sobre el proyecto, me gustaría destacar su complejidad aunque si todos los miembros del equipo hubiéramos llevado al día las clases de prácticas y con una mejor organización, la complejidad se hubiera reducido considerablemente.

Por último, en cuanto a la distribución de la carga de trabajo, considero que se nos ha acumulado para el final teniendo una importante carga de trabajo para esta última semana.

Jorge

Me ha parecido una idea de proyecto muy interesante para poder realizarla. Es una idea que bien desarrollada y llevada a cabo, puede tener una aceptación enorme en el mercado de las redes sociales. Las posibilidades que se pueden llegar a plantear son

infinitas. Sin embargo, en nuestro proyecto nos hemos limitado a llevar a cabo las principales funcionalidades que se nos pedían.

Personalmente me ha gustado mucho el funcionamiento de Angular y las posibilidades que ofrece de cara al desarrollo web, puesto que una vez comprendes como funciona, desarrollar interfaces con funcionalidad me parece más sencillo que nunca. Poder dotar de funcionalidad a esas plantillas de bootstrap que ya había visto en más de una ocasión es algo que me da cierto orgullo respecto a la asignatura.