Informe de Avance Twitter

Integrantes:

Ray Díaz Vega - Fabian Hernandez - Carlos Villalobos - Arnulfo Torres

Introducción

Las redes sociales representan una gran parte de la vida cotidiana de las personas hoy en día. Algunos las utilizan simplemente para comunicarse con otras personas y ver el estado de sus amigos y familiares, mientras que otros la utilizan como método de mercadeo o venta de productos. En cualquiera de los casos podemos ver que se encuentra una utilidad para calcular estadísticos que nos permitan ver el impacto que tiene la actividad de los usuarios sobre los demás.

Un ejemplo de cómo podría ser útil ver el impacto de tus tweets en Twitter o posts en Facebook es ver a qué tipo de público están atrayendo tus publicaciones y así adaptarte para seguir atrayendo más y más audiencia a tu cuenta. Por esta razón, se decidió crear un microservicio que permita visualizar la información de una cuenta de Twitter, sus últimos 5 tweets y cierta información acerca de los mismos, incluyendo la eficiencia (impacto generado en base a los seguidores del usuario) de cada tweet.

Todo esto se puede realizar utilizando nuevas tecnologías capaces de facilitar el manejo de información que transita actualmente en la nube proporcionando facilidad de uso y accesos como tal por medio de API, las API (Interfaces de programación de aplicaciones) no son más que un conjunto de comandos, funciones y protocolos informáticos que permiten a los desarrolladores crear programas específicos para ciertos sistemas operativos. Las API simplifican en gran medida el trabajo de un creador de programas, ya que no tiene que «escribir» códigos desde cero. Estas permiten al informático usar funciones predefinidas para interactuar con el sistema operativo o con otro programa.

Objetivos

- Desarrollar una aplicación que permita ver estadísticas relacionadas con el impacto de una cuenta de red social en sus seguidores.
- Crear una API que permita visualizar cierta información acerca de una cuenta de Twitter y sus últimas cinco publicaciones.
- Conectar la API con los demás microservicios de la aplicación a través del Middleware y hacer que se pueda visualizar con el Dashboard.

Metodología Scrum

Se decidió utilizar esta metodología, porque provee un marco de trabajo de procesos ágiles que trabaja con el ciclo de vida iterativo e incremental, donde se va liberando el producto por pares de forma periódica, aplicando las buenas prácticas de trabajo colaborativo (en equipo), facilitando el hallazgo de soluciones óptimas a los problemas que pueden ir surgiendo en el proceso de desarrollo del proyecto.

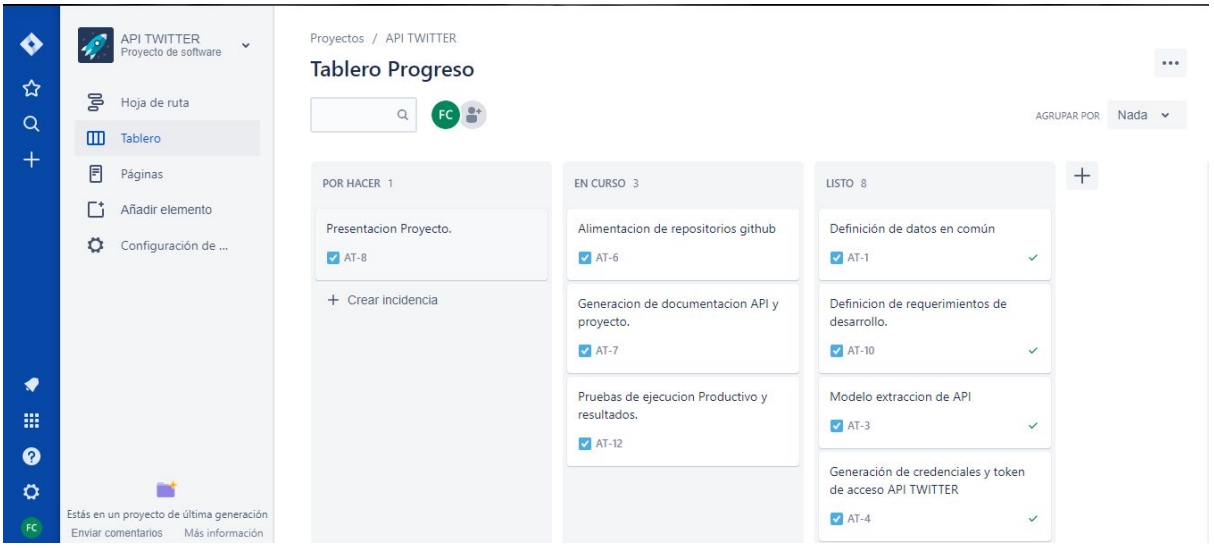
Con Scrum se realizan entregas regulares y parciales (sprint) del producto final, todas ellas con una prioridad previamente establecida que nace según el beneficio que aporten al cliente, minimizando los riesgos que pueden surgir de desarrollos extremadamente largos. Es por tal motivo, que Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesitan obtener resultados de manera inmediata y donde son fundamentales los siguientes aspectos: la innovación, la productividad, la flexibilidad y la competitividad.

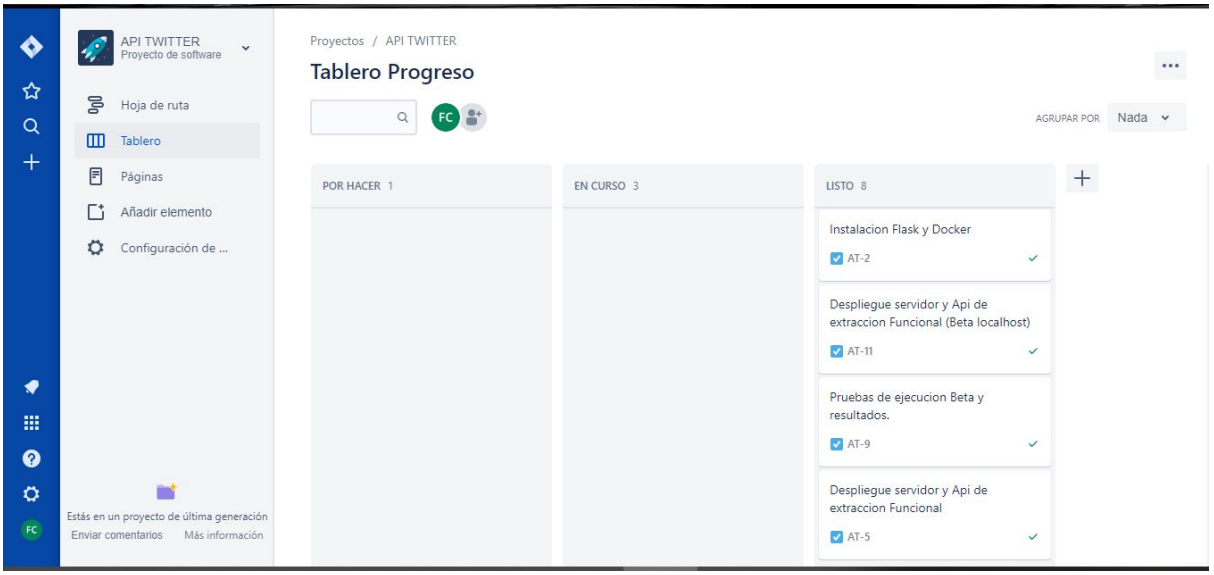
Stakeholder: Jairo serrano (Cliente)

Product Owner: Grupo F Arquitectura software

Scrum Master: Grupo F Arquitectura software

Scrum Team (Equipo de desarrollo): Grupo F Arquitectura software





https://proyecto211.atlassian.net/jira/software/projects/AT/boards/1_URL DE SEGUIMIENTO.

Utilización de Tweepy

Se estableció que todos los microservicios de la aplicación (Reddit, Youtube, Instagram...) deberían utilizar la misma información de las cuentas para calcular la eficiencia de las publicaciones. Por lo tanto, decidimos utilizar librería Tweepy de Python, que nos permite acceder a la API de Twitter y extraer la información necesaria para calcular el impacto de los tweets. Los datos que se utilizaron se dividen en dos; la cuenta que se está consultando y las publicaciones de dicha cuenta. Cada uno de estos datos se despliegan de la siguiente manera:

Cuenta

- *Nombre*

  Nombre completo de la persona que utiliza la cuenta.
- *Followers*

  Cantidad de seguidores de la cuenta de Twitter.
- *ID*

  Número entero único con el que se puede identificar la cuenta de Twitter.
- *Número de Posts*

  Cantidad de posts que ha hecho el usuario de la cuenta.
- *Username*

  Nombre con el que se identifica la cuenta de Twitter a consultar.

Posts (Tweets)

- *fecha*

  Día, mes, año y hora en que se publicó el tweet.
- *eficiencia*

  La eficiencia marca el impacto que tuvo un tweet en las personas que lo vieron, es decir, qué tanta atención logró capturar dicha publicación. Para lograr ver la eficiencia de los tweets de una cuenta en Twitter, se utilizó la siguiente fórmula:

$$Eficiencia = \frac{\#\ de\ favoritos\ del\ post}{\#\ de\ followers\ del\ usuario} * 100$$

- *likes*

  Número de favoritos que obtuvo el tweet.
- *id*

  Número de identificación único del tweet.
- *usuario*

  Nombre del usuario que hizo la publicación.

Cabe recalcar que para hacer uso de la API de Twitter es necesario tener una credencial de desarrollador. Esta credencial se puede generar fácilmente a través de la página https://developer.twitter.com/. Una vez se haya creado una aplicación y generado las llaves de autenticación, estas pueden ser utilizadas para usar el microservicio sin problema.

Condiciones de uso.

Para la utilización de cualquier clase de api que se quiera consumir para un proyecto o aplicación como tal se deben tener en cuentas algunas cláusulas o condiciones establecidas por parte de la entidad desarrolladora de dicha API, en este caso twitter tiene ciertas condiciones establecidas para la API utilizada en este proyecto capaz de extraer información necesaria para nuestro proyecto de medición de eficiencia por las publicaciones que puede realizar un usuario en específico, a continuación enunciamos algunas condiciones que están establecidas por parte del creador de la API (Twitter) a utilizar:

Puedes:

- Desarrollar aplicaciones que transmitan de forma automática información en los Tweets, mientras no incumplas las normas de Tweets automatizados.
- Publicar campañas de creatividades que responden automáticamente a los usuarios que interactúan con tu contenido.
- Desarrollar soluciones que responden a los usuarios de forma automática en los Mensajes Directos.
- Probar cosas nuevas que ayudan a la gente (y que cumplen con nuestras reglas).
- Asegurarte de que tu aplicación ofrezca una buena experiencia de usuario y de que tenga un buen rendimiento, y constata que siga siendo así a lo largo del tiempo.

No puedes:

- Incumplir estas u otras políticas. Ten en cuenta especialmente nuestras reglas sobre las prácticas abusivas y la privacidad del usuario.
- Abusar de la API de Twitter o intentar eludir los límites de velocidad.
- Enviar spam o importunar a los usuarios, o bien enviarles cualquier tipo de mensajes no solicitados.

Diagrama de funcionalidad API

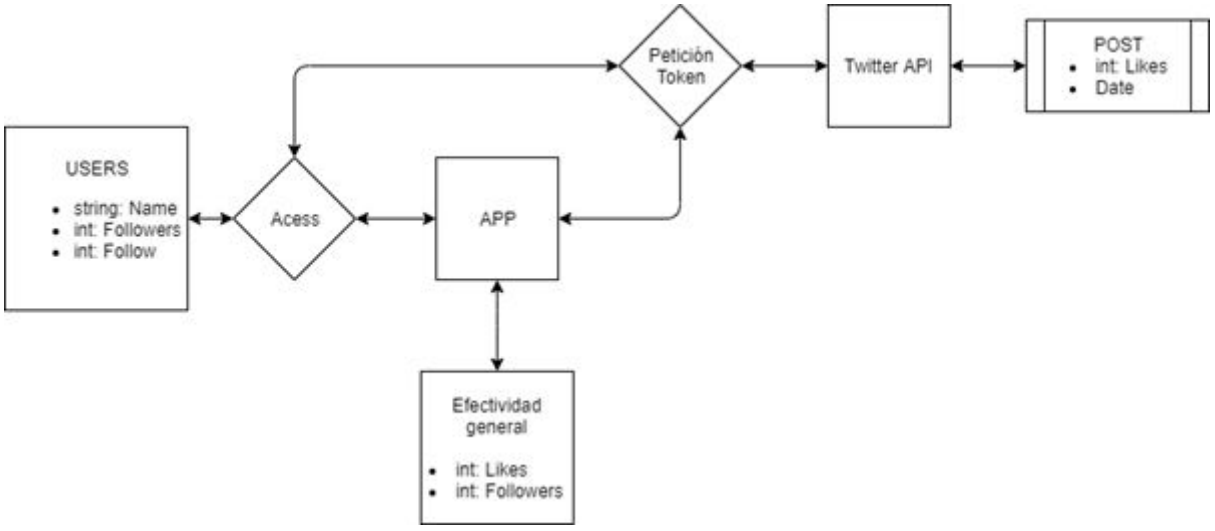
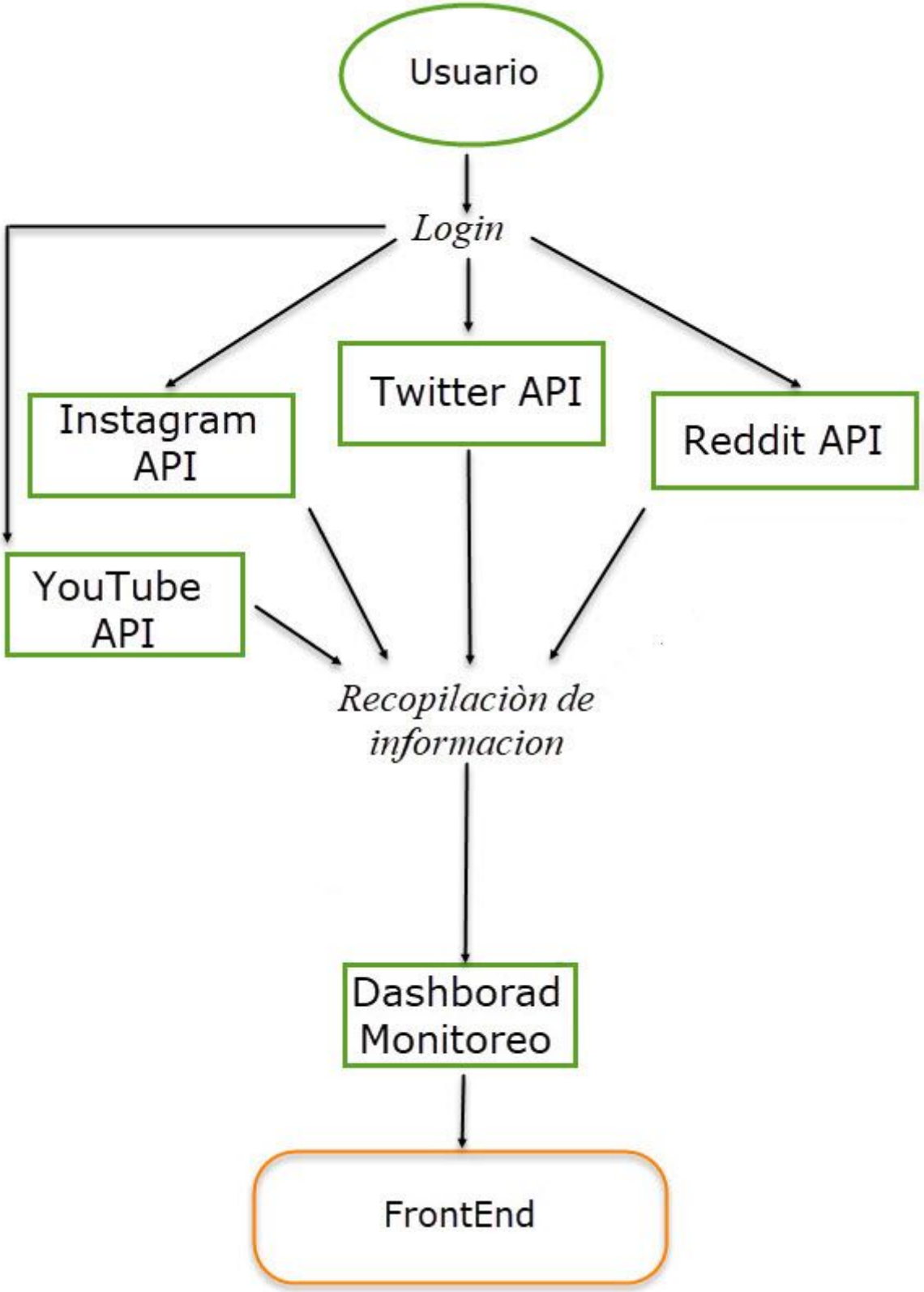
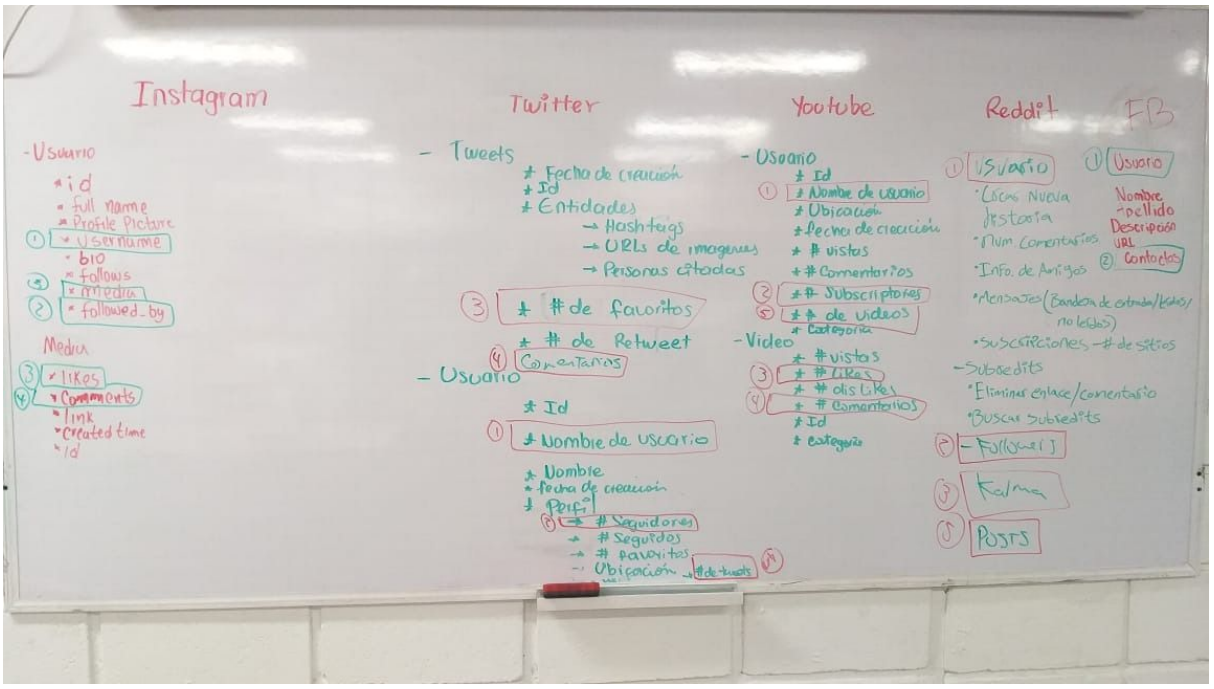


Diagrama de diseño conjunto de APIs



Estudio de datos en común por API



Requerimientos de desarrollo

FLASK

Para el desarrollo de esta aplicación capaz de extraer información específica y realizar cálculos u operaciones específicas como la eficiencia de cada post que realice un usuario fue necesaria la utilización de un framework que permita una usabilidad y adaptación de manera sencilla y eficaz a las necesidades planteadas para estas tareas, capaz de permitir un desarrollo claro y ordenado de los requerimientos y procesos a implementar, para el presente caso hemos optado por la utilización del micro frameWork FLASK para nuestro lenguaje de desarrollo a utilizar Python que posee como objetivo mantener el núcleo simple pero extensible. Flask no tomará muchas decisiones por usted, como qué base de datos usar. Esas decisiones que toma, como qué motor de plantillas usar, son fáciles de cambiar.

Todo lo demás depende de usted, para que Flask pueda ser todo lo que necesita y nada que no necesite.

Se necesitan unas dependencias básicas o requerimientos como tal para poder utilizar este macro framework las cuales enunciamos a continuación en conjunto a su proceso de instalación para su uso:

Entornos virtuales

se debe utilizar un entorno virtual para administrar las dependencias de su proyecto, tanto en desarrollo como en producción.

¿Qué problema resuelve un entorno virtual? Cuantos más proyectos de Python tenga, es más probable que necesite trabajar con diferentes versiones de las bibliotecas de Python, o incluso con Python. Las versiones más nuevas de las bibliotecas para un proyecto pueden romper la compatibilidad en otro proyecto.

Los entornos virtuales son grupos independientes de bibliotecas de Python, uno para cada proyecto. Los paquetes instalados para un proyecto no afectarán a otros proyectos ni a los paquetes del sistema operativo.

Python 3 viene incluido con el módulo **venv** para crear entornos virtuales.

- **Como crear un entorno**

Cree una carpeta de proyecto y una carpeta `venv` dentro de:

```
$ mkdir myproject
$ cd myproject
$ python3 -m venv venv
```

En Windows:

```
$ py -3 -m venv venv
```

Si necesita instalar virtualenv porque está utilizando Python 2, use el siguiente comando en su lugar:

$ python2 -m virtualenv venv

En Windows:

> \Python27\Scripts\virtualenv.exe venv

- **Activar el entorno**

Antes de trabajar en su proyecto, active el entorno correspondiente:

$ .venv/bin/activate

En Windows:

> venv\Scripts\activate

Su indicador de shell cambiará para mostrar el nombre del entorno activado.

- **Instalar Flask**

Dentro del entorno activado, use el siguiente comando para instalar Flask:

$ pip install Flask.

Una vez ejecutado el comando Flask quedará instalado en su computadora listo para trabajar.

Docker

Docker es una herramienta diseñada para facilitar la creación, implementación y ejecución de aplicaciones mediante el uso de contenedores. Los contenedores permiten a un desarrollador empaquetar una aplicación con todas las partes que necesita, como bibliotecas y otras dependencias, y enviarla como un paquete.

Para instalar Docker seguimos la guía de instalación oficial.
https://docs.docker.com/v17.09/engine/installation/

- **Cree una imagen de nuestro Dockerfile con el siguiente comando**

Docker build --tag = image_name.

- **Implemente la aplicación con el comando**
-

docker run -p 8081: 80 nombre_imagen

- **Abra el navegador, en la barra de navegación coloque http: // localhost: 8081 y veremos las rutas disponibles.**

Usuario: / usuario /

Publicaciones: / publicaciones /

Referencias

- Flask information
- API conceptos basicos
- Tweepy Org
- Twitter Developers
- Python Org
- IngenieriaDeSistemasUTB/ArcSoft2p2019 (GitHub Project)
- Docker