**APIs *A****pplication* ***P****rogramming* ***I****nterface*

“una aplicación (intermediario) que realiza comunicaciones entre “nosotros” y una herramienta/aplicación/Web (endpoint)”

***REST API:***

* HTTP : la rest api son una manera de **realizar comunicaciones client - server:***una solicitud del navegador del cliente al servidor(endpoint) que le devuelve una respuesta*
* “Stateless”: llamadas individualizadas
* “App layer”
* Request & Response (enviamos una petición y nos devuelve una respuesta “satisfactoria” o no)
* **Request:** tienen los siguientes methods:
  + post “te doy información”
  + **get** “dame información”
  + put
  + delete
* **Response:** la respuesta

→ En el master trabajaremos con **REST API > REQUEST > GET** para solicitar información

**ELEMENTOS:**

* **endpoint** (<https://api.github>) + **resource/ruta** (/repos)
* **query string parameters** (?state=open) → info que nos queremos traer
* ***headers***: aporta info complementaria como los tokens de autenticación)

→ **REST API > REQUEST > GET > URL**

****

****

**PASOS:**

1. IMPORTAR LIBRERÍA:

**→ import requests**



1. OBTENER LA INFO DE LA API:

→ requests**.get(** url, {headers} **)**



1. PARSEARLO A UN STRING O JSON:

**→ .content()** ò **.json()** ò **.json\_normalize()**



1. tratar la información

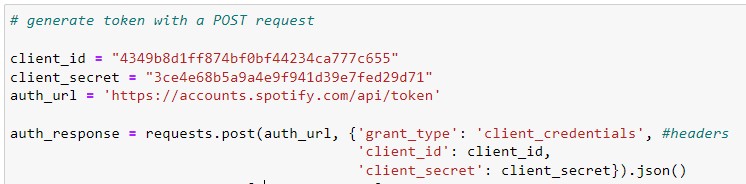


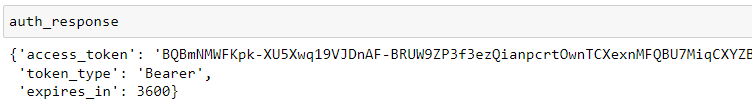
**\* .headers** → Nos devuelve un DIC con la info de los headers



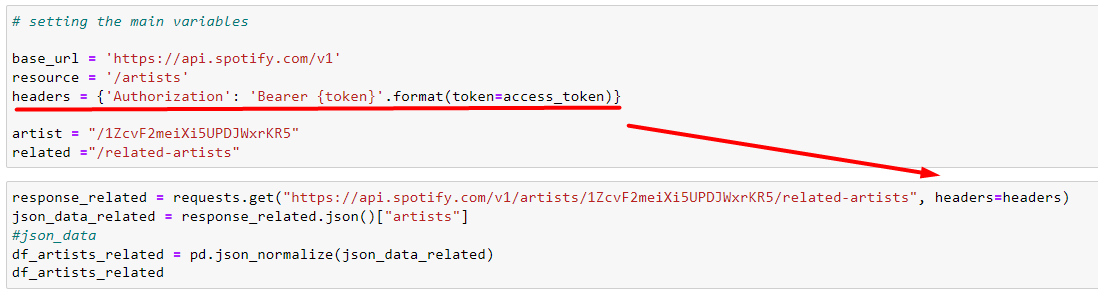
**EJEMPLO de requests con HEADERS**:

* De una request post para crear un token

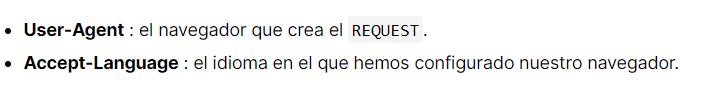




* De una request get para solicitar la info



\* También podemos simular User-Agent





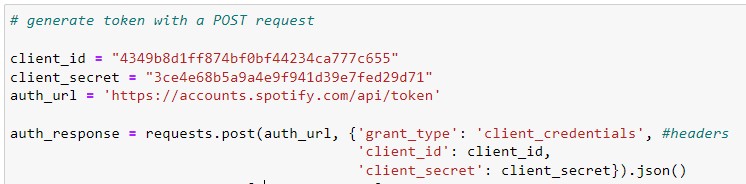
**CÓDIGOS:**

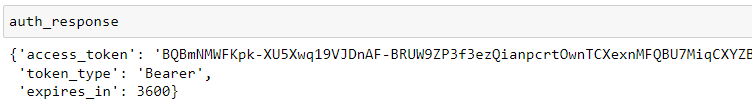
Todas las respuestas siguen la misma estructura. Tienen un HEADER que contiene metadatos sobre cuál es el comportamiento del servidor y un BODY que contiene la información deseada.

* 2xx Success: el código de estado más importante aquí es 200, que significa que **todo funcionó como se esperaba** y el recurso se encuentra en el archivo SERVER.
* Errores del cliente 4xx: Son códigos de **errores del lado del cliente**, lo que significa que el cliente solicitó un recurso incorrecto o, el que todo el mundo sabe, 404 que significa que el recurso **no se puede encontrar en el SERVER**.
* Errores del servidor 5xx: esto significa que el SERVER mismo se corrigió al procesar su REQUEST. Quizás debido a un **error en el código del servidor** o algo relacionado.

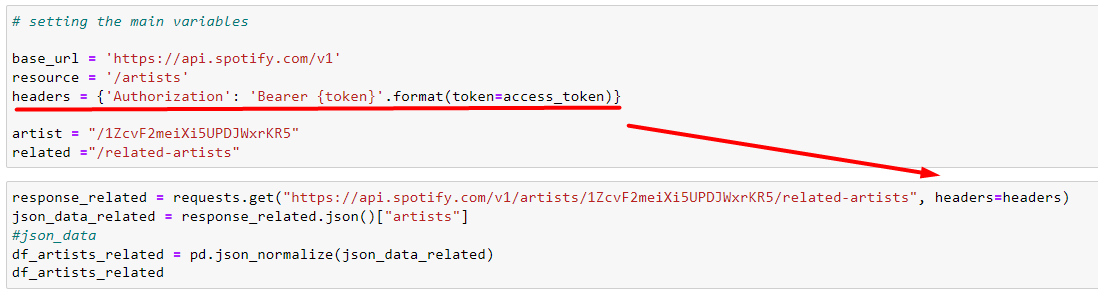
**Autenticación OAuth:**

Cuando se trabaja con OAuth, el servicio que están pidiendo una REQUEST deberá proporcionar algunas credenciales autenticadas que normalmente tienen nombres como **clientId**, **apiKey** y **apiSecret**, **apiToken** o simplemente **token**.





→ esta solicitud nos devuelve un token que usaremos en nuestras siguientes llamadas



**SPOTIFY:**

* **app-name:** spotify-alvaro-ironhack
* **Client ID:** 4349b8d1ff874bf0bf44234ca777c655
* **Client Secret:** 3ce4e68b5a9a4e9f941d39e7fed29d71

**Twitter:**

[**https://docs.tweepy.org/en/stable/api.html**](https://docs.tweepy.org/en/stable/api.html)

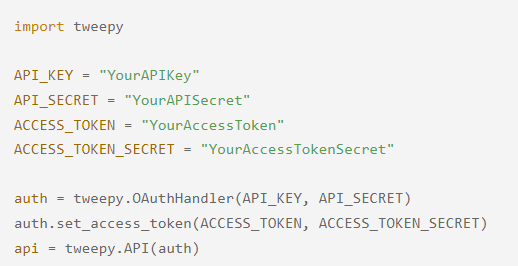
* **app-name:** ironhack\_alv
* **API Key:** BwJM1UhLWrJjD4c5IJc4JAOJB
* **API Key Secret:** mgYy8VkJg15b26aqTQeLbOQ4a4XTuzXe9GYZNToSUSthCO4RSS
* **Bearer Token**: AAAAAAAAAAAAAAAAAAAAANYBXAEAAAAAvOMCSaVPs2piRfYnW0BIYRCEpzY%3Dpkjz3cOwg1JsHjrHg7r1CSQBxgLWeAgg4w2MNp0lU1AKiV2cLz
* **Access Token:** 829330535663202306-VNTWjxUTslxZvwEClgN6fjdAruumBGo
* **Access Token Secret:** okQijywH0bSpzgDemHfTDcDJ89UwBvjIBYc5l9AXZBDZU

API DE TWITTER <https://docs.tweepy.org/en/stable/api.html>

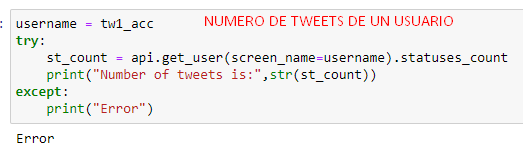
URL UTIL PARA SCRAPPING DE TWITTER: <https://www.geeksforgeeks.org/extraction-of-tweets-using-tweepy/?ref=lbp>

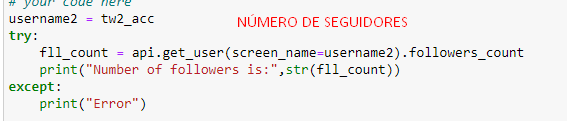
URL UTIL 2 PARA SCRAPPING DE TWITTER: <https://towardsdatascience.com/how-to-scrape-tweets-from-twitter-59287e20f0f1>

**Twitter tweepy:**









**Twitter snscrape:**

* [**https://medium.com/dataseries/how-to-scrape-millions-of-tweets-using-snscrape-195ee3594721**](https://medium.com/dataseries/how-to-scrape-millions-of-tweets-using-snscrape-195ee3594721)
* **https://medium.com/swlh/how-to-scrape-tweets-by-location-in-python-using-snscrape-8c870fa6ec25**
* GitHub containing this tutorial’s scraping files: <https://github.com/MartinBeckUT/TwitterScraper/tree/master/snscrape/>
* snscrape GitHub: <https://github.com/JustAnotherArchivist/snscrape>
* Medium Article for scraping tweets using CLI and wrapper:

<https://betterprogramming.pub/how-to-scrape-tweets-with-snscrape-90124ed006af>

* NOTEBOOK CON LOS PASOS QUE HAGO MAS ADELANTE:

<https://github.com/MartinBeckUT/TwitterScraper/blob/master/snscrape/python-wrapper/snscrape-python-wrapper.ipynb>

* query search avanzadas en twitter: https://github.com/igorbrigadir/twitter-advanced-search

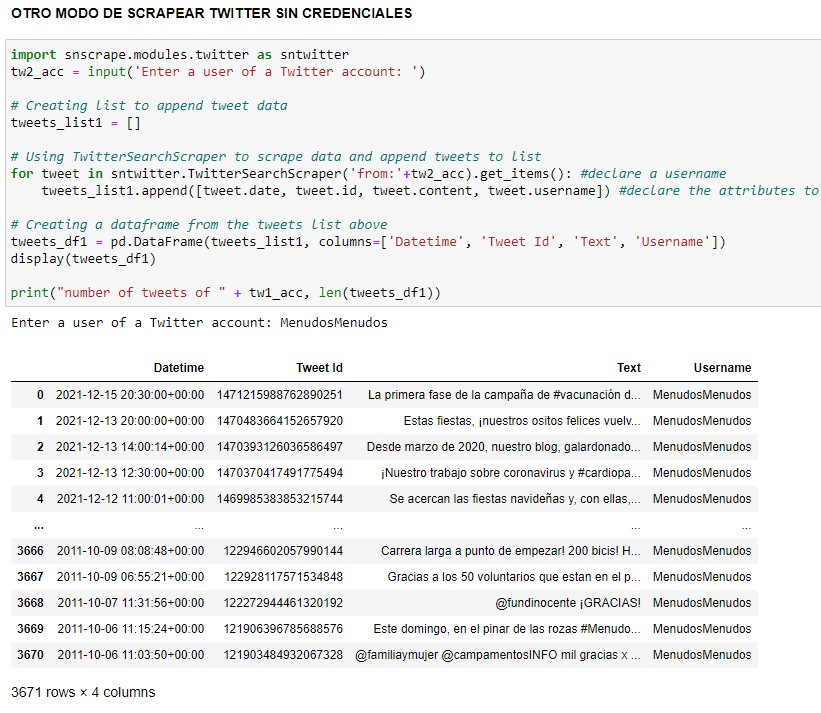
pip3 install snscrape

pip3 install git+<https://github.com/JustAnotherArchivist/snscrape.git>



1. buscando tweets por usuario





**\* SI QUEREMOS OBTENER UN NÚMERO LIMITADO DE TWEETS**



1. buscando búsquedas (CON INTERTOOLS ES COMO SI ITERÁSMEOS Y ESCOGIESEMOS UN NUMERO X DE TWEETS)

