Extracción de la Información de la Web

Máster Online en Ciencia de Datos

UCO ONLINE

Dr. José Raúl Romero

Profesor Titular de la Universidad de Córdoba y Doctor en Ingeniería Informática por la Universidad de Málaga. Sus líneas actuales de trabajo se centran en la democratización de la ciencia de datos (*Automated ML* y *Explainable Artificial Intelligence*), aprendizaje automático evolutivo y analítica de software (aplicación de aprendizaje y optimización a la mejora del proceso de desarrollo de software).

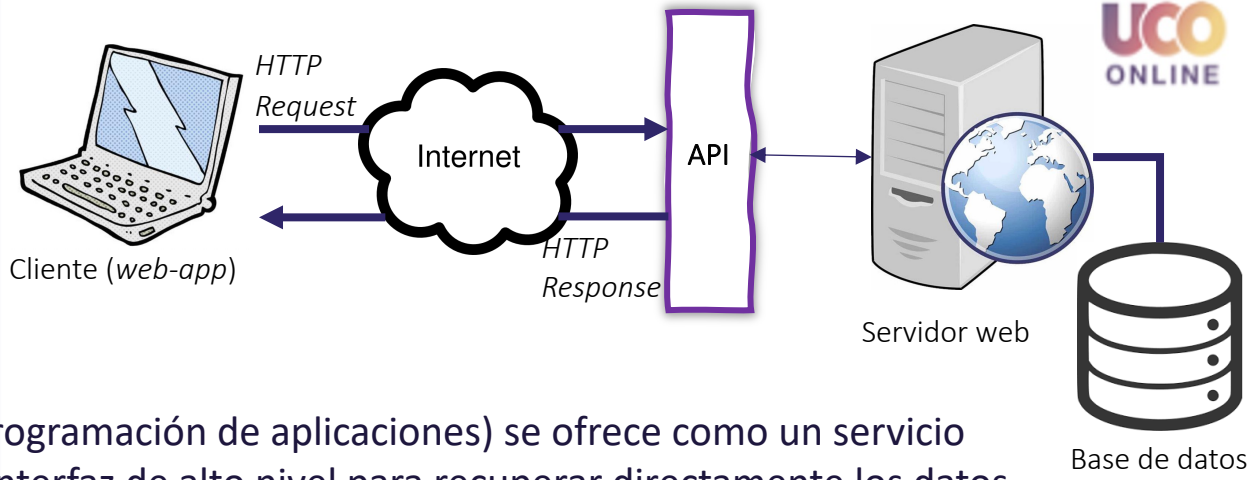
Miembro del Consejo de Administración de la *European Association for Data Science*, e investigador senior del Instituto de Investigación Andaluz de *Data Science and Computational Intelligence*.

Director del **Máster Online en Ciencia de Datos** de la Universidad de Córdoba.

Programación de acceso a través de APIs: el ejemplo de Twitter

UCO ONLINE

Definición



- Una API (interfaz de programación de aplicaciones) se ofrece como un servicio que proporciona una interfaz de alto nivel para recuperar directamente los datos de los repositorios o bases de datos alojados en el *backend* de un sitio Web
- Una API es "un conjunto de especificaciones, como los mensajes *HTTP Request*, junto con una definición de la estructura de los mensajes de respuesta, normalmente en un formato XML o JSON" (Wikipedia)
- Los accesos se realizan a través de puntos finales de URL (*URL endpoints*)
- Un estilo popular de arquitectura es REST (o transferencia de estado representacional), que permite la interacción con los servicios web a través de llamadas GET y POST de HTTP

Definición

- Por ejemplo, la API REST de Twitter permite a los desarrolladores acceder a los datos principales de Twitter y la API de búsqueda proporciona métodos para que los desarrolladores interactúen con los datos de búsqueda y tendencias de Twitter
- Existen principalmente dos formas de utilizar las API:
  - A través de envoltorios específicos del lenguaje de programación
  - A través del terminal de comandos utilizando puntos finales de URL

  Por ejemplo, **Tweepy** es una famosa envoltura de python para la API de Twitter, mientras que **twurl** es una herramienta de interfaz de línea de comandos (CLI)
- Una de las ventajas de utilizar las API oficiales es que suelen cumplir los términos de servicio (ToS) de un servicio concreto
  - ¡OJO! Los paquetes de terceros que afirman proporcionar un mayor rendimiento que las API oficiales (límites de velocidad, número de solicitudes/seg) pueden estar violando los ToS

Web scraping Vs. APIs: Diferencias

- El *scraping* web se centra en la recuperación de información específica que puede proceder de varios sitios web
  - El código del *scraper* se encarga de tomar los voluminosos datos no estructurados en datos en un formato estructurado
  - El *scraper* accede a todos los datos disponibles en la web
- La API dependen de un único sitio web, propietario del conjunto de datos (pago Vs. gratis, solicitudes limitadas o no, datos limitados o no, etc.)
  - Los datos se recuperan en un formato estructurado
  - La API accede a todos los datos que el propietario hace disponibles de la base de datos
- El *scraper* depende de servidores proxy, lo que no ocurre con la API
- El *web scraping* es mucho más personalizable, complejo y tiene un conjunto de reglas más prefijado.

Web scraping Vs. APIs: Ventajas del *scraping*

- Ambos métodos permiten recopilar datos de clientes e información que antes no se veían
- El *Web Scraping* puede ser más ventajoso para la extracción de datos:
  - Si se trata de una empresa que requiere información actualizada, el *web scraping* es mejor opción: es personalizable para extraer el tipo específico de información que se demanda
  - Ausencia de límites de velocidad: en las API existen limitaciones que el *web scraping* no tiene (en sentido técnico)
  - Las API pueden costar una fortuna y pueden resultar pesadas para las pequeñas empresas que buscan obtener inteligencia de mercado
  - En *web scraping* no habrá precio para extraer datos PERO es aconsejable no rastrear sitios web cuyo robot.txt lo advierta explícitamente
  - Datos limitados disponibles públicamente a través de una AP, por lo que se deberá utilizar el *web scraping* igualmente

Web *scraping* Vs. APIs: Ventajas del *scraping*

- No hay personalización con la API: frecuencia, formato y estructura, agente de usuario
- No todos los sitios web permiten el *scraping* de datos. En este caso, el uso de la API puede ser su única opción (p.ej. Facebook)
- Datos casi en tiempo real y relevantes: Las bases de datos de los sitios web obtenidas mediante API no pueden actualizarse en tiempo casi real, lo que hace que los datos queden obsoletos.
- Anonimato en el *web scraping*, que no es posible cuando se utiliza la API (habitualmente hay que registrarse y recibir una clave)
- Mejor estructuración en el *web scraping*: la estructura de la API no siempre es intuitiva (depende de los desarrolladores del sitio externo), mientras que extraer la estructura del sitio con XPath suele ser más sencillo inspeccionando el código de la página

Programación de acceso a través de APIs: el ejemplo de Twitter

Ejemplos

UCO ONLINE

https://pypi.org/project/wptools/

API para extracción de datos de Wikipedia

- **wptools** es una API de Python muy básica para extracción de datos de Wikipedia
- Se basa en la API de MediaWiki (motor de Wikipedia)

  https://www.mediawiki.org/wiki/API:Main_page/es

- Permite navegar por páginas de Wikipedia (con un título o id concreto, o aleatorias), extraer páginas, imágenes, estadísticas de las páginas, etc.

API para extracción de datos de Twitter

- **Tweepy** es una librería de Python para extracción de datos de Twitter
- Utiliza la API de Twitter, que permite el acceso a los elementos principales de la plataforma: Tweets, Mensajes Directos, Espacios, Listas, usuarios y más.

https://developer.twitter.com/en/docs/twitter-api

- El acceso a Twitter API se gestiona desde el portal del desarrollador de Twitter

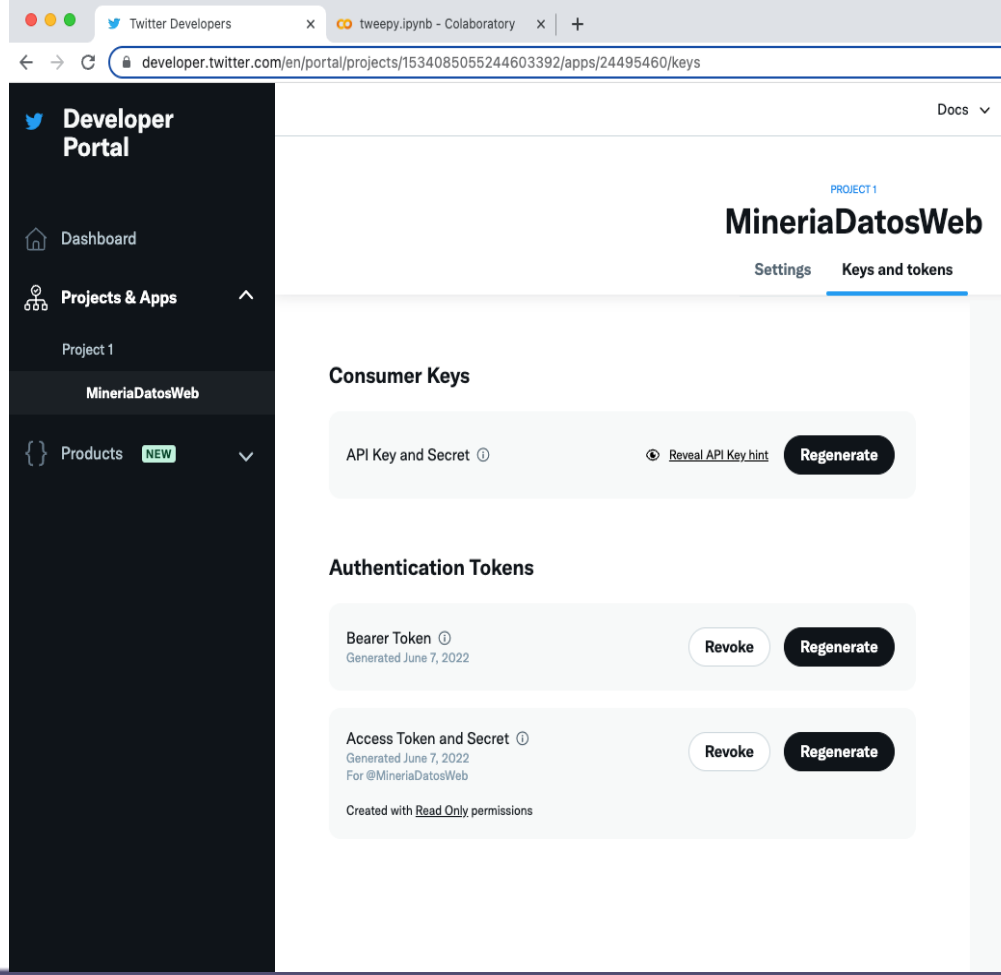
https://developer.twitter.com/en/docs/developer-portal/overview

Credenciales y autenticación

- La API de Twitter requiere que el programador obtenga credenciales de autenticación (*tokens*) que deben pasarse en cada solicitud:
  - ***API Key and Secret***: esencialmente el nombre de usuario y la contraseña de su aplicación. Se utiliza para autenticar las solicitudes que requieran el *OAuth 1.0a User Context*, o para generar otros tokens como los *Access Tokens* o los *App Access Tokens*
  - ***User Access Tokens***: en general, los *Access Tokens* representan al usuario que está haciendo la petición en su nombre. Se generan a través del portal de desarrolladores y representan al usuario propietario de la aplicación. Se utilizan para autenticar solicitudes que requieran el *OAuth 1.0a User Context*.
  - ***App Access Token***: se Utiliza este token para hacer una solicitud a un *endpoint* que requiera *OAuth 2.0 App-Only*

Credenciales y autenticación

- Al registrarnos, obtenemos el *API_key* y *API_secret*
- Una vez creada una aplicación, necesitamos entrar en el dashboard del desarrollador y generar el *access_token* y *access_token_secret*
  - Entramos en la aplicación concreta
  - Solapa "*Keys and tokens*"
  - En *Authentication Tokens* > *Generate* (*Access Token and Secret*)



Programación con Tweepy

- Veamos ejemplos de uso de Tweepy
- Genera tus credenciales para API Twitter y toma el Bearer Token
- Accede al cuaderno Google Colab (Canvas) y personaliza el Bearer

¡Comenzamos!

¡Gracias!

UCO ONLINE