

BIG DATA Y PYTHON

MÁSTER EN BIG DATA

201020

GABRIEL MARÍN DÍAZ

hola

Presentación

Yo mismo

Nombre: Gabriel Marín Díaz

A qué me dedico...

- Channel Enablement Manager en Sage
- Profesor Asociado UCM

Perfil de LinkedIn: <https://www.linkedin.com/in/gabrielmarindiaz/>

CONTENIDO

Contenido

Resumen

Tema 1 – Visión General

Tema 2 – Introducción a SQL

Tema 3 – Introducción al Lenguaje Python

Tema 4 – HTML y Python

Tema 5 – Big Data y Python

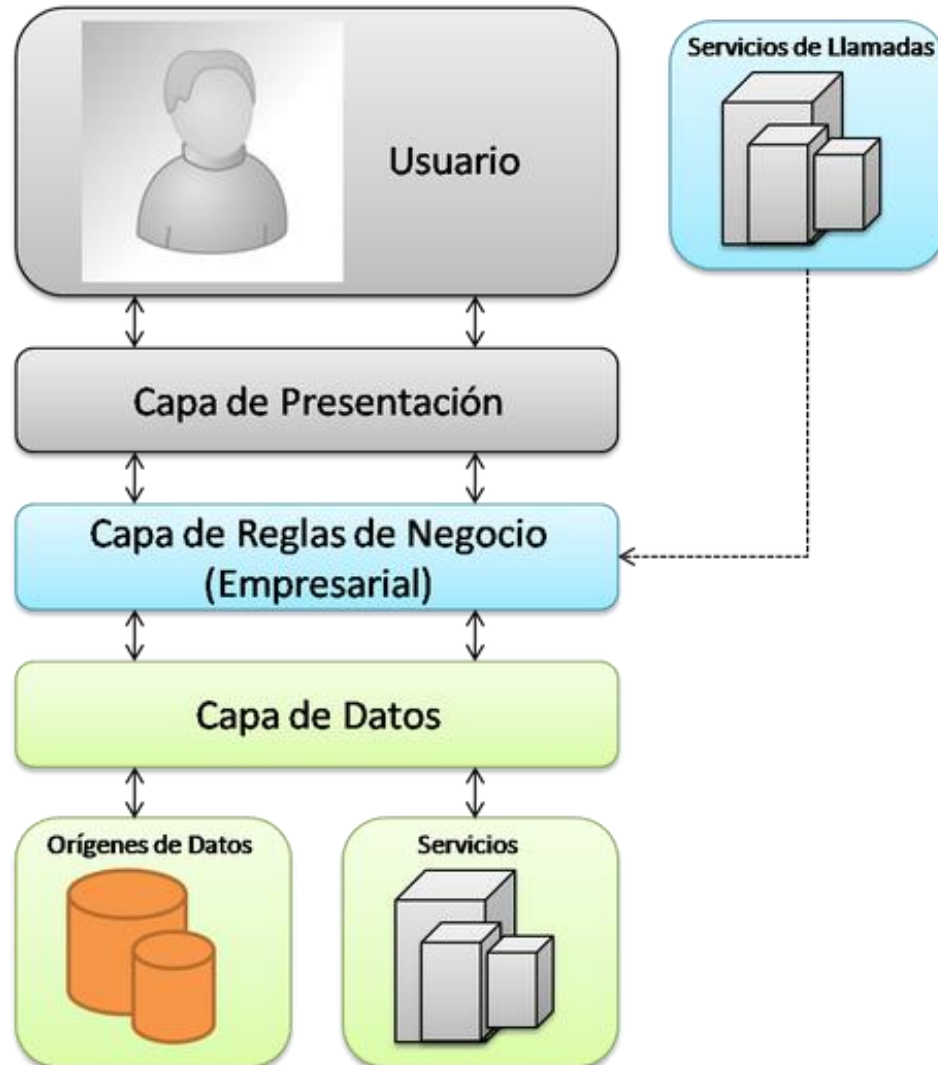
Tema 6 – Procesamiento Distribuido (Spark)

Prácticas las realizaremos con Python, MySQL, MongoDB, Apache Spark

Arrancamos?

CONTEXTO

Arquitectura en capas



1. Capa de Presentación: Interacción entre el usuario y el software. Puede ser tan simple como un menú basado en líneas de comando o tan complejo como una aplicación basada en formas. Su principal función es mostrar información al usuario, interpretar los comandos de este y realizar algunas validaciones simples de los datos ingresados.

2. Capa de Reglas de Negocio (Empresarial): También denominada Lógica de Dominio, esta capa contiene la funcionalidad que implementa la aplicación. Involucra cálculos basados en la información dada por el usuario, datos almacenados y validaciones. Controla la ejecución de la capa de acceso a datos y servicios externos.

3. Capa de Datos: Esta capa contiene la lógica de comunicación con otros sistemas que llevan a cabo tareas por la aplicación. Para el caso de aplicaciones empresariales, está representado por una base de datos, que es responsable del almacenamiento persistente de información. Esta capa debe abstraer completamente a las capas superiores (negocio) del dialecto utilizado para comunicarse con los repositorios de datos (PL/SQL, Transact-SQL, etc.).

Índice

- ☐ Lectura de Ficheros
- ☐ Web Scraping
- ☐ Uso de APIs
- ☐ Mongo DB
- ☐ Procesamiento Distribuido con SPARK
- ☐ Visualización de Resultados

CONTINUAMOS CON PYTHON...
WEB SCRAPING

Uso de APIs

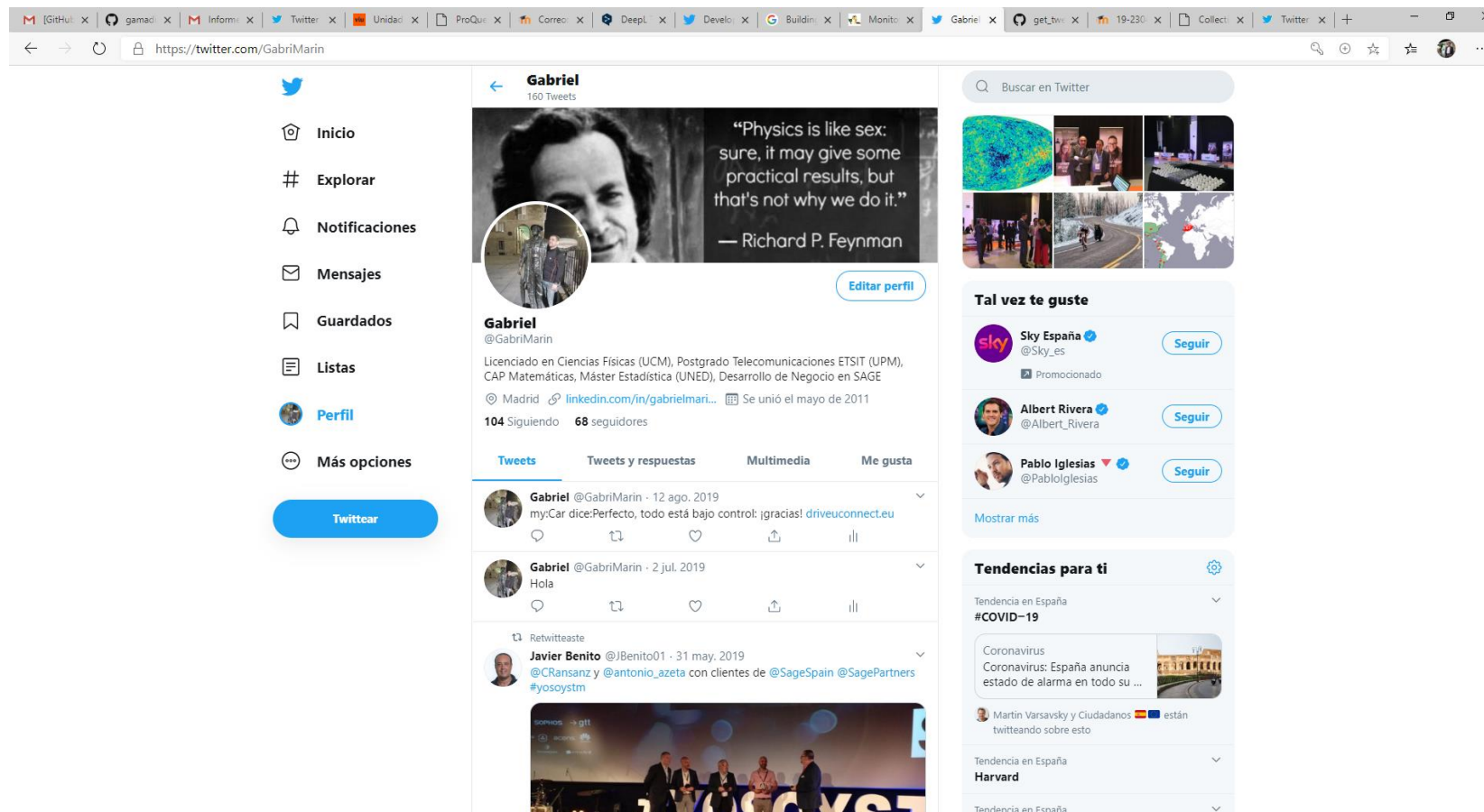
En las páginas en las que los datos son más ricos o cambian más habitualmente no tiene sentido seguir el modelo de descarga de datos, sería necesario actualizar el fichero todos los días. Veremos dos APIs...

- a) El API de Twitter (tweepy), descargaremos los mensajes que nos interesen, ya sea consultando los mensajes ya existentes o bien, “escuchando” según se emiten (streaming).
- b) El API-REST de OMDB, la mayor parte de las redes sociales disponen de su propia API-REST. En cada caso consultaremos la información del servicio web al que queramos acceder para conocer los parámetros que requiere.

Uso de APIs

API TWITTER

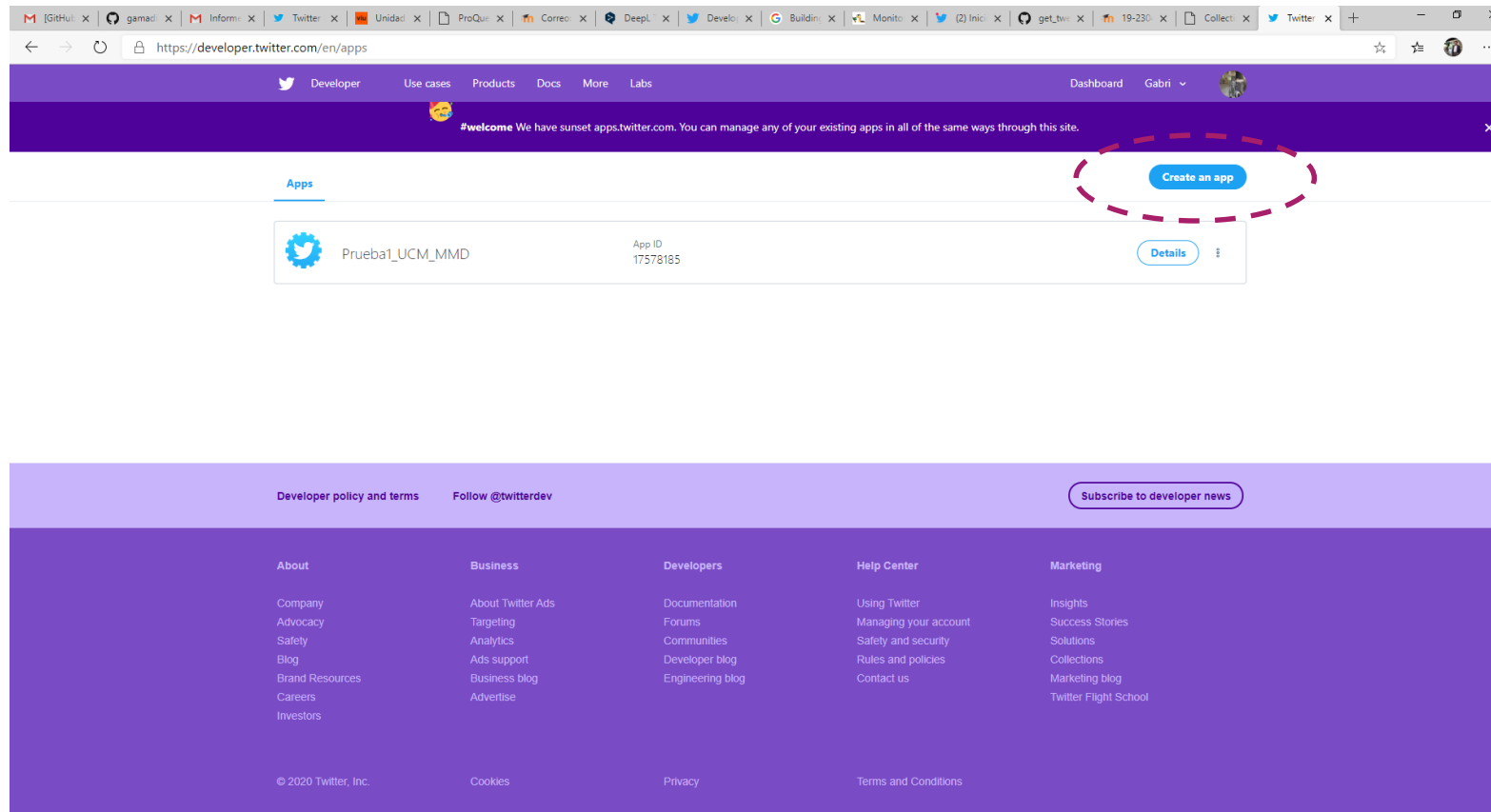
Suponemos que tenemos una cuenta en Twitter, si no la tenéis por favor creadla para hacer estas pruebas.



Uso de APIs

API TWITTER

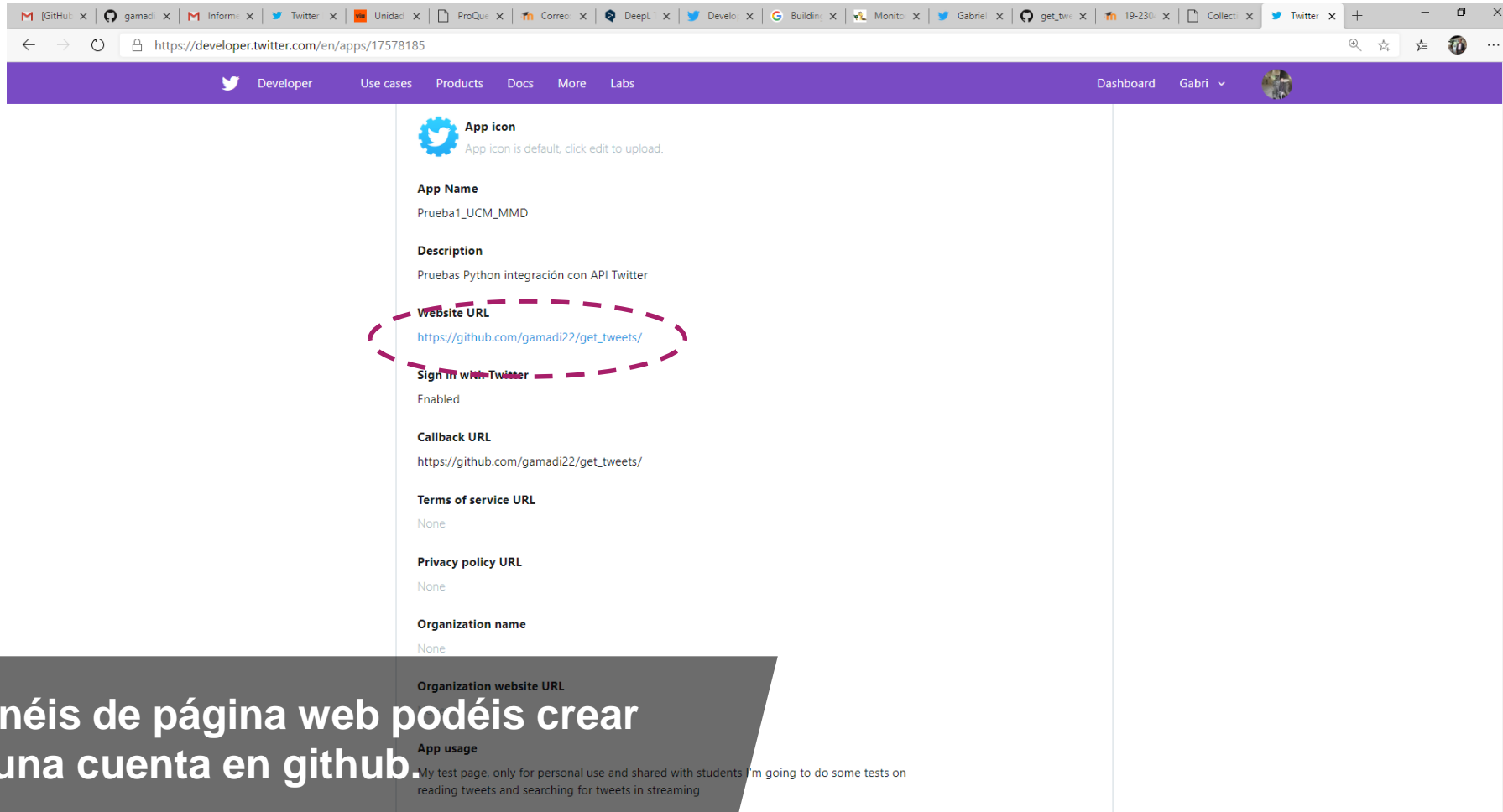
Una vez creada la cuenta, accederemos a <https://apps.twitter.com>, indicando que queremos crear una nueva aplicación. Twitter tarda un poco en habilitar el perfil desarrollador, así que solicitarlo cuanto antes para poder hacer las pruebas.



Uso de APIs

API TWITTER

Twitter nos solicitará información acerca de los detalles de la aplicación.



The screenshot shows the Twitter Developer Portal configuration page for an application. The 'Website URL' field is highlighted with a red dashed circle. The form includes fields for App Name, Description, Website URL, Sign in with Twitter, Callback URL, Terms of service URL, Privacy policy URL, Organization name, and Organization website URL. The 'App usage' section at the bottom contains a text area with the following content:

My test page, only for personal use and shared with students I'm going to do some tests on reading tweets and searching for tweets in streaming

Si no disponéis de página web podéis crear una cuenta en github

Para conseguir todos los token necesarios para descargar tweets iremos a la pestaña “Keys and Access Tokens” y pulsaremos el botón “Create my access token”

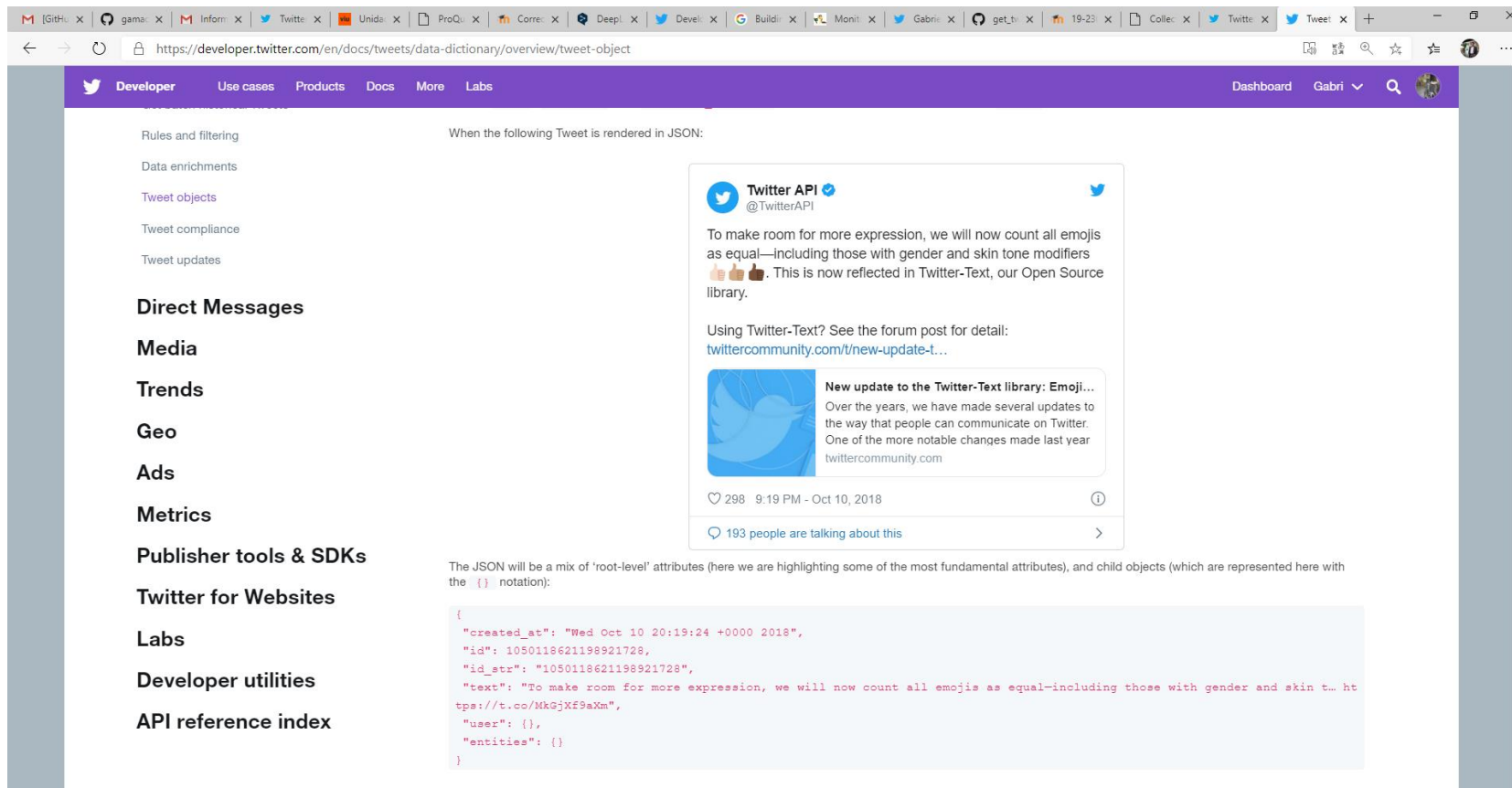
The screenshot shows the Twitter Developer portal interface. At the top, there's a navigation bar with tabs: 'Developer', 'Use cases', 'Products', 'Docs', 'More', and 'Labs'. The 'Developer' tab is active. Below this, there's a breadcrumb trail: 'Apps > Prueba1_UCM_MMD'. Underneath, there are three tabs: 'App details', 'Keys and tokens' (highlighted with a red dashed circle), and 'Permissions'. The main content area is titled 'Important notice about your access token and access token secret'. Below this, there's a section titled 'Keys and tokens' with a subtitle 'Keys, secret keys and access tokens management'. This section contains two sub-sections: 'Consumer API keys' and 'Access token & access token secret'. The 'Consumer API keys' section shows an 'API key' and an 'API secret key'. The 'Access token & access token secret' section shows an 'Access token' and an 'Access token secret'. A red dashed circle highlights the 'Keys and tokens' tab in the navigation bar.

Uso de APIs

API TWITTER

Estructura de un tweet

Los tweets que descargamos con Tweepy son objetos JSON con la siguiente estructura:



The screenshot shows the Twitter Developer API documentation page for the tweet object. The page title is "When the following Tweet is rendered in JSON:". The main content area displays a tweet from the Twitter API (@TwitterAPI) about counting emojis as equal. Below the tweet, the JSON structure is shown, highlighting the 'root-level' attributes and child objects.

Twitter API
@TwitterAPI

To make room for more expression, we will now count all emojis as equal—including those with gender and skin tone modifiers 🍌👉👉👉. This is now reflected in Twitter-Text, our Open Source library.

Using Twitter-Text? See the forum post for detail: twittercommunity.com/t/new-update-t...

New update to the Twitter-Text library: Emoji...
Over the years, we have made several updates to the way that people can communicate on Twitter. One of the more notable changes made last year twittercommunity.com

298 9:19 PM - Oct 10, 2018

193 people are talking about this

The JSON will be a mix of 'root-level' attributes (here we are highlighting some of the most fundamental attributes), and child objects (which are represented here with the {} notation):

```
{
  "created_at": "Wed Oct 10 20:19:24 +0000 2018",
  "id": 1050118621198921728,
  "id_str": "1050118621198921728",
  "text": "To make room for more expression, we will now count all emojis as equal—including those with gender and skin t... ht",
  "user": {},
  "entities": {}
}
```

Click me!

Uso de APIs

API TWITTER

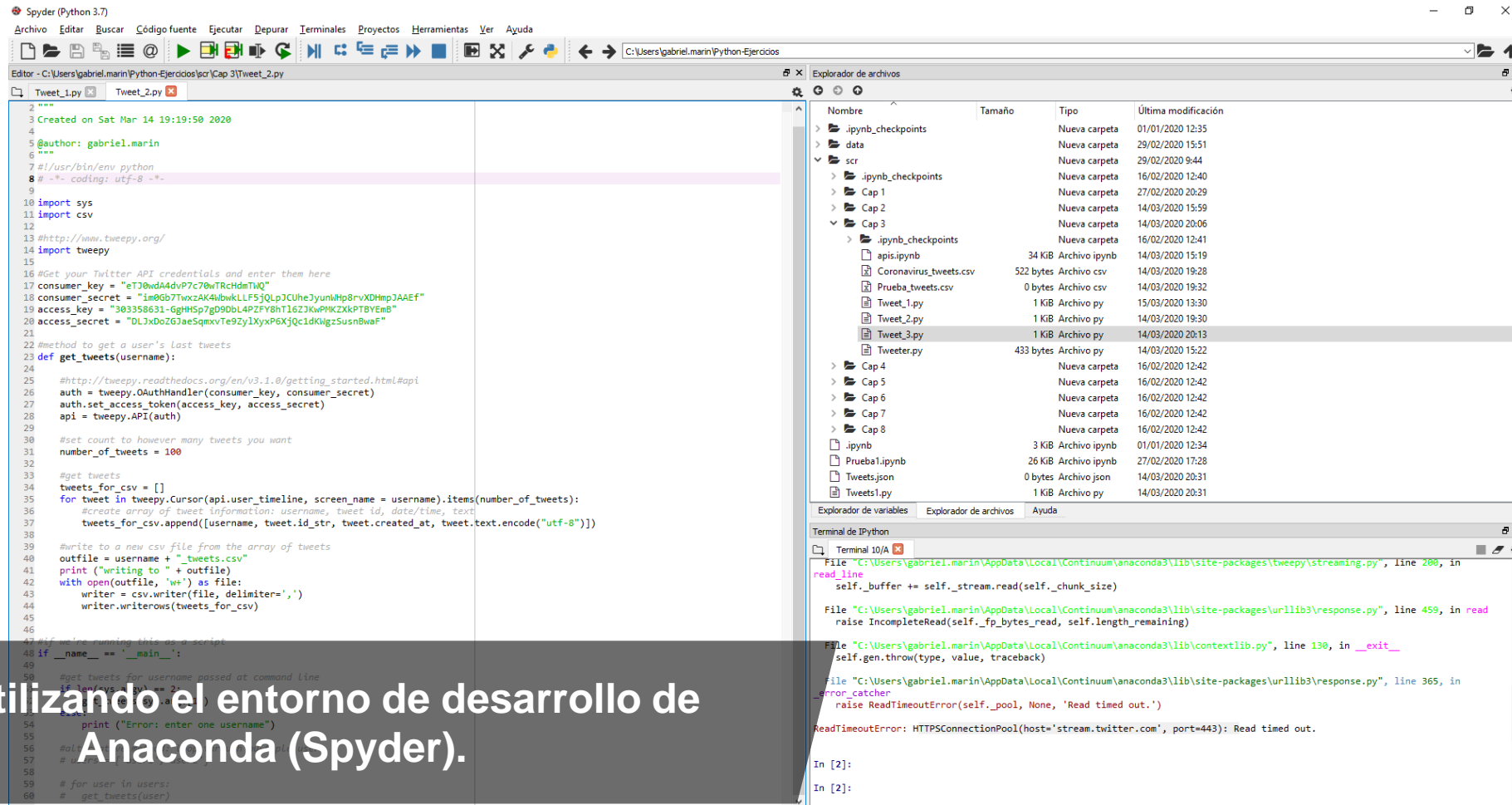
Descargando tweets

Existen dos formas de descargar tweets:

- A. Buscando tweets almacenados.
- B. Escuchando continuamente (streaming) lo que se publica y guardando aquellos que cumplan ciertos criterios.

NOTA: no obtendremos todos los tweets ya que existen limitaciones de pago en la cantidad de información a la que podemos acceder, en general se pueden acceder a un número mayor de tweets utilizando el método streaming.

Un ejemplo de búsqueda

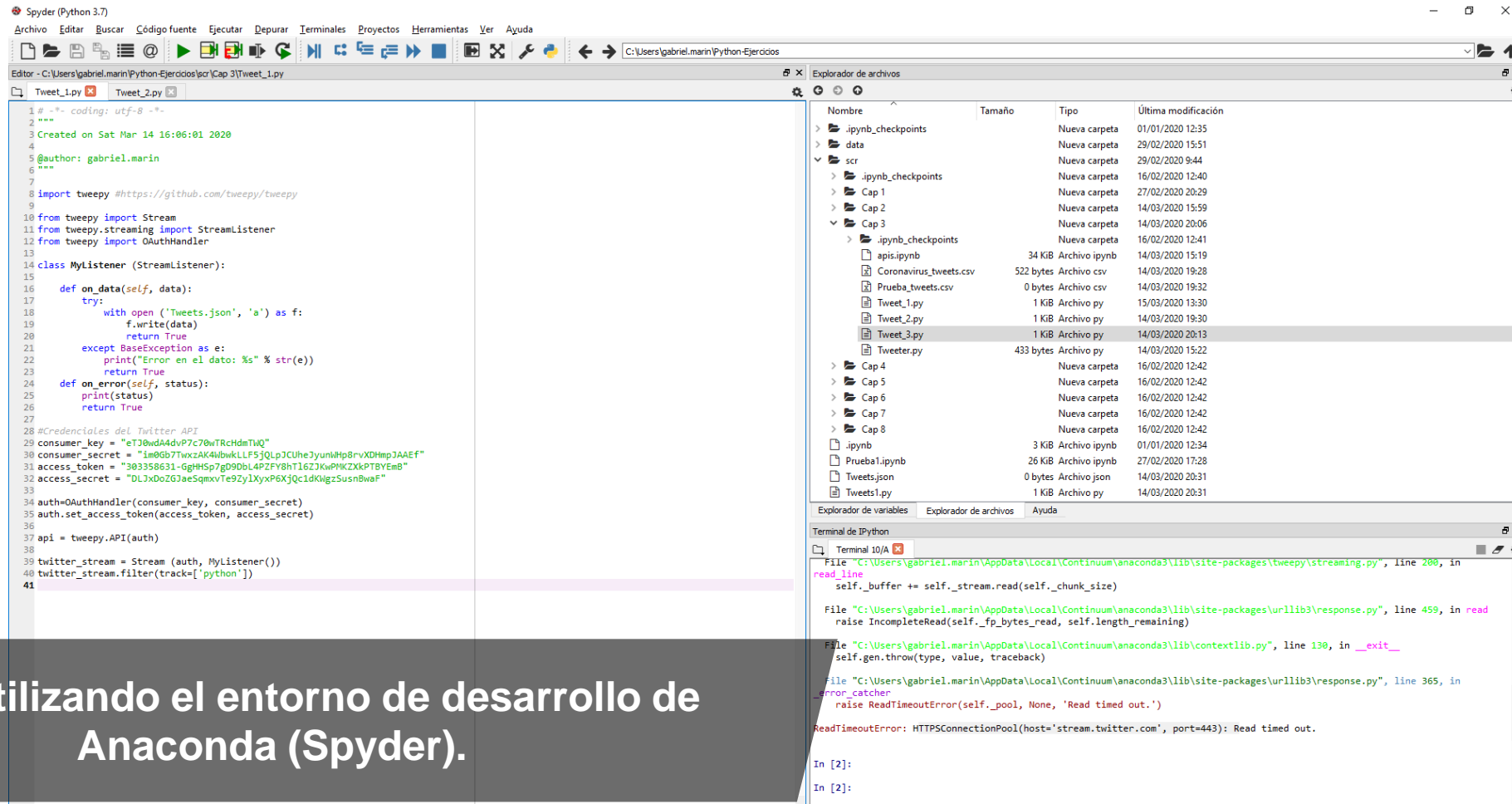


Click me!

Uso de APIs

API TWITTER

Un ejemplo de escucha



The screenshot shows the Spyder Python IDE interface. The main editor displays a Python script for listening to the Twitter API using Tweepy. The script includes imports for Tweepy, Stream, StreamListener, and OAuthHandler. It defines a MyListener class with on_data and on_error methods. The script also includes Twitter API credentials and initializes the listener. The file explorer on the right shows the project structure, including folders for checkpoints, data, and scripts, and files for API keys, tweets, and the listener script. The terminal window at the bottom shows a ReadTimeoutError: HTTPConnectionPool(host='stream.twitter.com', port=443): Read timed out.

```
1 # -*- coding: utf-8 -*-
2 """
3 Created on Sat Mar 14 16:06:01 2020
4
5 @author: gabriel.marin
6 """
7
8 import tweepy #https://github.com/tweepy/tweepy
9
10 from tweepy import Stream
11 from tweepy.streaming import StreamListener
12 from tweepy import OAuthHandler
13
14 class MyListener(StreamListener):
15
16     def on_data(self, data):
17         try:
18             with open('Tweets.json', 'a') as f:
19                 f.write(data)
20             return True
21         except BaseException as e:
22             print("Error en el dato: %s" % str(e))
23             return True
24     def on_error(self, status):
25         print(status)
26         return True
27
28 #Credenciales del Twitter API
29 consumer_key = "eTJ0wdA4dvP7c70wTRchdmTWQ"
30 consumer_secret = "im0G67TwxzAK4vbwkLLF5jQLpJCuHeJyunkHp8rvXDhmp3AAEF"
31 access_token = "303358631-6gHSp7g09DbL4PZFY8hT16ZJWpMKZKkPTBYEmB"
32 access_secret = "DLJxDoZGJaeSqmxvTe9ZyLXyxP6XJQc1dKlWgzSusnBwaF"
33
34 auth=OAuthHandler(consumer_key, consumer_secret)
35 auth.set_access_token(access_token, access_secret)
36
37 api = tweepy.API(auth)
38
39 twitter_stream = Stream(auth, MyListener())
40 twitter_stream.filter(track=['python'])
41
```

Terminal de IPython

```
File "C:\Users\gabriel.marin\AppData\Local\Continuum\anaconda3\lib\site-packages\tweepy\streaming.py", line 288, in read_line
    self._buffer += self._stream.read(self._chunk_size)
File "C:\Users\gabriel.marin\AppData\Local\Continuum\anaconda3\lib\site-packages\urllib3\response.py", line 459, in read
    raise IncompleteRead(self._fp_bytes_read, self.length_remaining)
File "C:\Users\gabriel.marin\AppData\Local\Continuum\anaconda3\lib\contextlib.py", line 130, in __exit__
    self.gen.throw(type, value, traceback)
File "C:\Users\gabriel.marin\AppData\Local\Continuum\anaconda3\lib\site-packages\urllib3\response.py", line 365, in _error_catcher
    raise ReadTimeoutError(self._pool, None, 'Read timed out.')
ReadTimeoutError: HTTPConnectionPool(host='stream.twitter.com', port=443): Read timed out.

In [2]:
In [2]:
```

Estoy utilizando el entorno de desarrollo de Anaconda (Spyder).



EJERCICIO

Uso de APIs

EJERCICIO 1

Vamos a mejorar el ejercicio de escucha en twitter...

- Introduzcamos como parámetro de escucha “COVID-19”.
- Escuchemos durante un período de tiempo razonable (30 minutos).
- El resultado obtenido lo dirigiremos a un fichero JSON (“escucha_covid.json”).
- Utilizando la librería de Python Pandas, vamos a ir creando un estudio de los tweets generados: ubicación del tweet, texto, fuente del tweet (source), número de respuestas que ha recibido el tweet,... y todo lo que se os ocurra para obtener estadísticas

Esto es un anticipo de lo que podrá ser la segunda práctica. Es importante que reviséis la estructura de un tweet, de ahí podréis obtener toda la información y que trabajéis la biblioteca Pandas para análisis de datos.

CONTINUAMOS...

Uso de APIs

API-REST

El protocolo API-REST define un método sencillo para recibir y enviar datos en cualquier formato, habitualmente XML o JSON, bajo el protocolo HTTP. Para ello se define un pequeño número de funciones para manipular la información, habitualmente POST, GET, PUT, DELETE.

Utilizaremos la biblioteca “*requests*”, ya utilizada para descargar ficheros en la clase anterior, y en particular la función “*get*”, que recibe como parámetro la dirección del recurso y un diccionario con las opciones de búsqueda y devuelven una respuesta (objeto de la clase “*response*”). Si los objetos que estamos manejando son de tipo JSON podemos solicitarle dicho objeto a la respuesta con la función “*json*”.

A continuación, un ejemplo...

EJERCICIO

Uso de APIs

EJERCICIO 2

Probemos con el API de Google Maps...

- Realizar un ejercicio que permita recoger la información de Google Maps.
- Daremos dos puntos inicio / final de un determinado trayecto.
- El API de Google Maps nos devolverá la distancia entre estos dos puntos.

Este ejercicio para trabajar en clase... Utilizaremos la biblioteca requests para obtener datos. Revisad el API de Google Maps en la siguiente dirección.

<https://developers.google.com/maps/documentation/directions/start?hl=es>

Como segunda parte del ejercicio deberéis proponer para la siguiente clase un ejercicio que permita utilizar las APIs de Google para obtener el camino más corto entre dos puntos seleccionando el medio de transporte (andando, tren, coche, avión,...).

Índice

- ☐ Lectura de Ficheros
- ☐ Web Scraping
- ☐ Uso de APIs
- ☐ Mongo DB
- ☐ Procesamiento Distribuido con SPARK
- ☐ Visualización de Resultados

VISUALIZACIÓN DE DATOS

(PARTE 1)

Visualización de Datos

Hasta ahora hemos aprendido a extraer información de distintas fuentes de datos, el objetivo es transformar datos en información para tomar decisiones.

El problema ahora consiste en cómo interpretar los resultados y cómo transmitir nuestras conclusiones a los demás.

Visualización de Datos

UTILIZAREMOS LA BIBLIOTECA **MATPLOTLIB**

Visualización de Datos

Existe una gran variedad de módulos para hacer gráficos de todo tipo con Python, pero el estándar de facto en ciencia es **matplotlib**. Se trata de un paquete grande y relativamente complejo que entre otros contiene dos módulos principales, **pyplot** y **pylab**.

pyplot ofrece una interfaz fácil para crear gráficos fácilmente, automatizando la creación de figuras y ejes automáticamente cuando hace un gráfico. Por otra parte, **pylab** combina la funcionalidad de **pyplot** para hacer gráficos con funcionalidad de **numpy** para hacer cálculos con arrays usando un único espacio de nombres muy parecido a **Matlab**.

Por esto, es posible que en la literatura nos encontremos dos formas comunes de usar la interfaz de matplotlib:



Click me!

Visualización de Datos

Representación con colores:

Colores

Símbolo	Color
"b"	Azul
"g"	Verde
"r"	Rojo
"c"	Cian
"m"	Magenta
"y"	Amarillo
"k"	Negro
"w"	Blanco

Marcas y líneas

Símbolo	Descripción
"_"	Línea continua
"-"	Línea a trazos
"-."	Línea a puntos y rayas
".."	Línea punteada
"."	Símbolo punto
"."	Símbolo pixel
"o"	Símbolo círculo relleno
"v"	Símbolo triángulo hacia abajo
"^"	Símbolo triángulo hacia arriba
"<"	Símbolo triángulo hacia la izquierda
">"	Símbolo triángulo hacia la derecha
"s"	Símbolo cuadrado
"p"	Símbolo pentágono
"s"	Símbolo estrella
"+"	Símbolo cruz
"x"	Símbolo X
"D"	Símbolo diamante
"d"	Símbolo diamante delgado

Click me!

Visualización de Datos

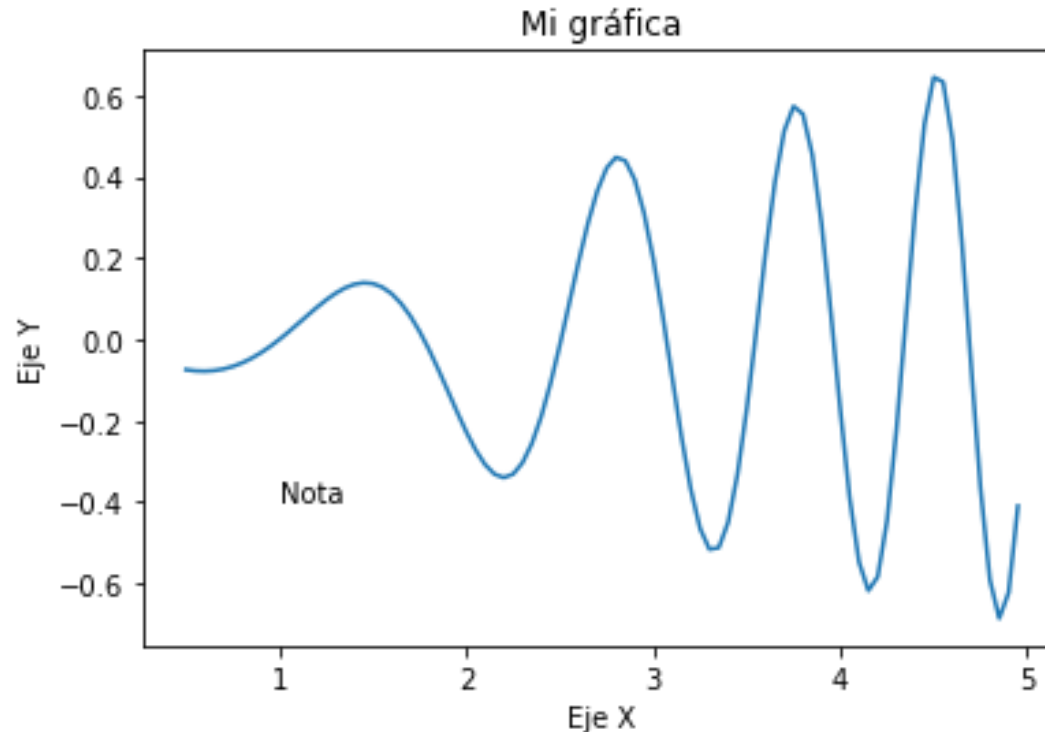
Además del marcador y el color indicado de la manera anterior, se pueden cambiar muchas otras propiedades de la gráfica como parámetros de `plot()` independientes como los de la tabla adjunta:

Parámetro	Significado y valores
<code>alpha</code>	grado de transparencia, float (0.0=transparente a 1.0=opaco)
<code>color</code> o <code>c</code>	Color de matplotlib
<code>label</code>	Etiqueta con cadena de texto, string
<code>markeredgecolor</code> o <code>mec</code>	Color del borde del símbolo
<code>markeredgewidth</code> o <code>mew</code>	Ancho del borde del símbolo, float (en número de puntos)
<code>markerfacecolor</code> o <code>mfc</code>	Color del símbolo
<code>markersize</code> o <code>ms</code>	Tamaño del símbolo, float (en número de puntos)
<code>linestyle</code> o <code>ls</code>	Tipo de línea, <code>"-"</code> <code>"--"</code> <code>"-."</code> <code>"."</code> <code>"None"</code>
<code>linewidth</code> o <code>lw</code>	Ancho de la línea, float (en número de puntos)
<code>marker</code>	Tipo de símbolo, <code>"+"</code> <code>"x"</code> <code>"o"</code> <code>"^"</code> <code>">"</code> <code>"<"</code> <code>"1"</code> <code>"2"</code> <code>"3"</code> <code>"4"</code> <code>"D"</code> <code>"H"</code> <code>"^"</code> <code>"_"</code> <code>"d"</code> <code>"h"</code> <code>"o"</code> <code>"p"</code> <code>"s"</code> <code>"v"</code> <code>"x"</code> <code>" "</code> TICKUP TICKDOWN TICKLEFT TICKRIGHT



Visualización de Datos

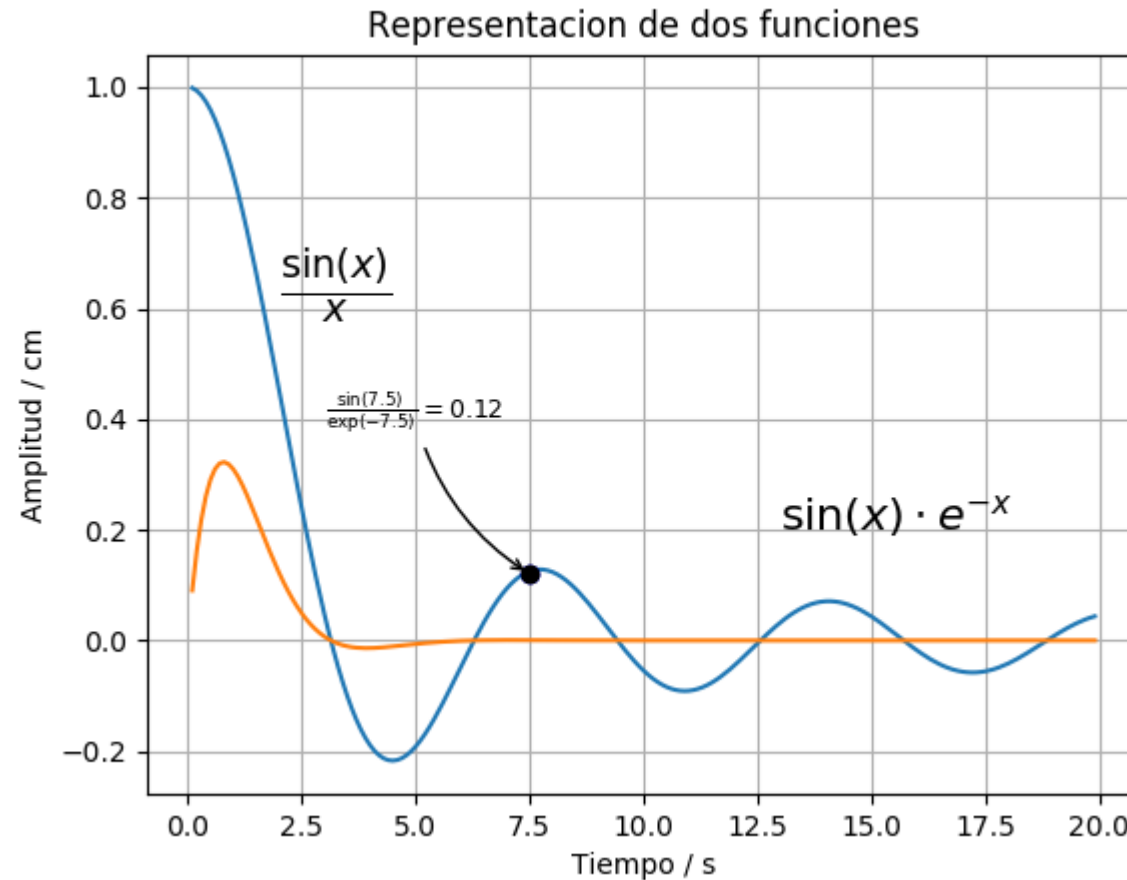
Existen funciones para añadir texto (etiquetas) a los ejes de la gráfica y a la gráfica en sí; éstos son los más importantes:



Click me!

Visualización de Datos

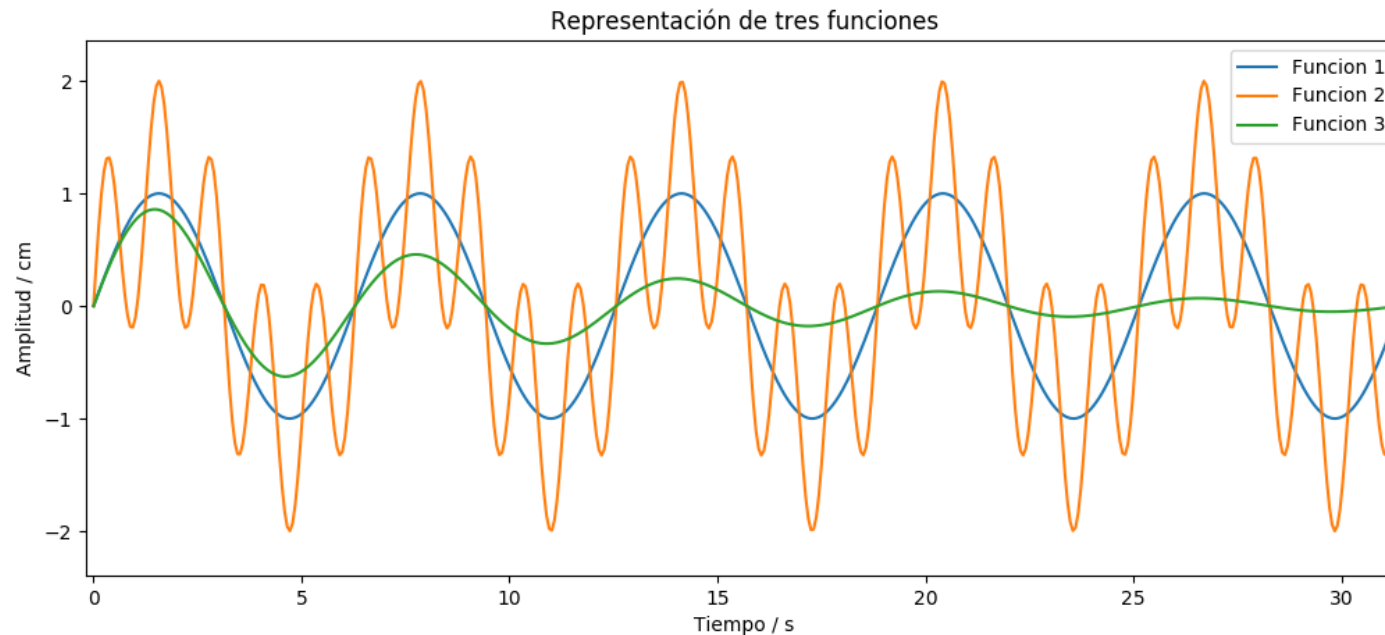
Cuando se utilizan textos también es posible usar fórmulas con formato LaTeX.



Click me!

Visualización de Datos

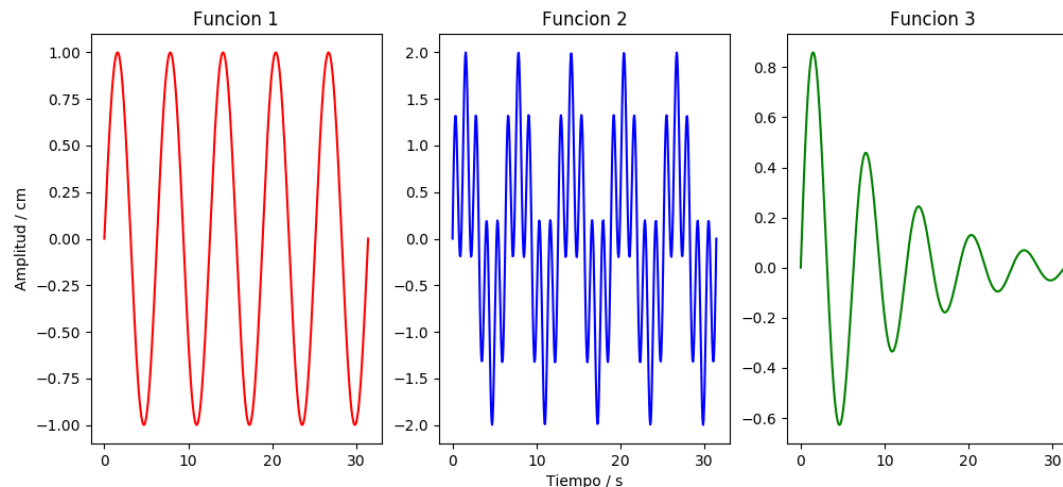
En Python es muy fácil representar gráficamente una función matemática. Para ello, debemos definir la función y luego generar un array con el intervalo de valores de la variable independiente que se quiere representar. Definamos algunas funciones trigonométricas y luego representémoslas gráficamente:



Click me!

Visualización de Datos

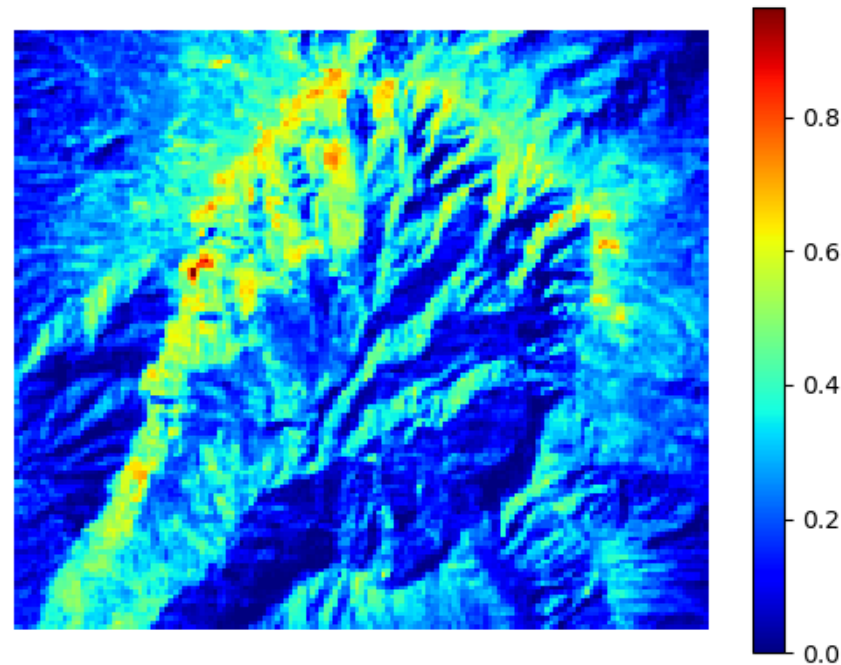
En ocasiones nos interesa mostrar varios gráficos diferentes en una misma figura o ventana. Para ello podemos usar la función **subplot()**, indicando entre paréntesis un número con tres dígitos. El primer dígito indica el número de filas en los que se dividirá la figura, el segundo el número de columnas y el tercero se refiere al gráfico con el que estamos trabajando en ese momento.



Click me!

Visualización de Datos

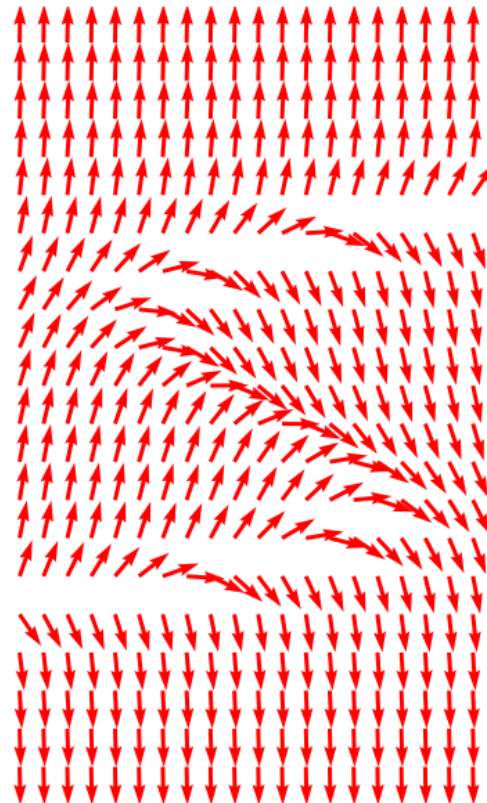
Los datos bidimensionales son valores de una magnitud física representada por una función que tiene dos variables independientes, normalmente x e y ; se trata pues de representar funciones del tipo $z = z(x,y)$, donde z puede representar flujo luminoso, presión atmosférica, altura del terreno, etc.



Click me!

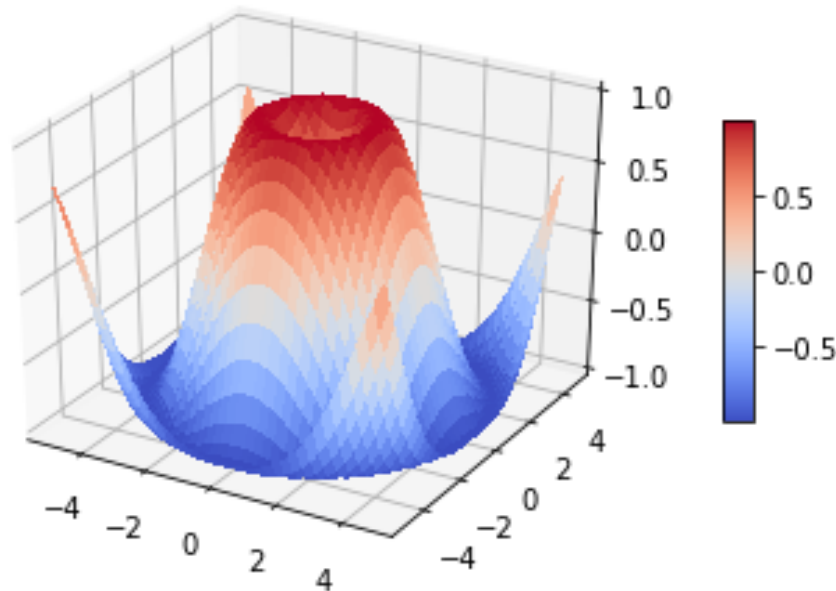
Visualización de Datos

Después de crear una figura con cualquiera de los procedimientos descritos hasta ahora podemos guardarla con la función `savefig()` poniendo como parámetro el nombre del fichero con su extensión.



Visualización de Datos

Aunque **matplotlib** está especializado en gráficos 2D, incluye un toolkit para hacer gráficos 3D de muchos tipos usando OpenGL, que nos resolverá casi todas las necesidades para gráficos de este tipo.



Click me!

EJERCICIO 1

Visualización de Datos

Vamos a utilizar los datos de desempleo empleo proporcionados para el mes de marzo de 2018 por entonces el Ministerio de Empleo y Seguridad Social.

Autoguardado: parosexoedadprov - Modo de compatibilidad - Excel

Marín Díaz, Gabriel

Archivo Inicio Insertar Disposición de página Fórmulas Datos Revisar Vista Ayuda Power Pivot Sage ERP X3

Buscar Compartir Comentarios

Portapapeles Fuente Alineación Número Estilos Celdas Edición Ideas Commands Group Jira Cloud

P14

Servicio Público de Empleo Estatal

Paro registrado según sexo y edad por provincias y comunidades autónomas

MARZO DE 2018

	TOTAL EDAD			MENORES DE 25 AÑOS		DE 25 A 29 AÑOS		DE 30 A 44 AÑOS		MAYORES DE 45 AÑOS	
	TOTAL	HOMBRES	MUJERES	HOMBRES	MUJERES	HOMBRES	MUJERES	HOMBRES	MUJERES	HOMBRES	MUJERES
ALMERIA	58.373	25.853	32.520	2.402	2.435	2.579	3.744	8.321	12.567	12.551	13.774
CADIZ	156.715	64.859	91.856	6.604	6.361	6.877	9.256	20.806	33.761	30.572	42.478
CORDOBA	75.099	31.997	43.102	3.658	3.892	3.457	5.043	9.292	15.044	15.590	19.123
GRANADA	86.366	39.420	46.946	4.028	4.090	4.291	5.779	12.485	17.967	18.616	19.110
HUELVA	46.853	21.140	25.713	2.105	1.840	2.103	2.702	6.459	9.488	10.473	11.683
JAEN	53.212	21.912	31.300	2.682	3.045	2.723	4.074	6.439	10.866	10.068	13.315
MALAGA	156.149	65.596	90.553	6.702	6.427	6.291	8.576	20.324	32.205	32.279	43.345
SEVILLA	204.659	86.615	118.044	8.872	8.831	8.952	11.990	27.687	43.430	41.104	53.793
ANDALUCIA	837.426	357.392	480.034	37.053	36.921	37.273	51.164	111.813	175.328	171.253	216.621
HUESCA	8.850	3.822	5.028	519	432	367	489	1.187	1.950	1.749	2.157
TERUEL	5.922	2.473	3.449	305	329	211	338	719	1.204	1.238	1.578
ZARAGOZA	54.935	22.089	32.846	2.721	2.532	1.950	2.840	6.524	11.328	10.894	16.146
ARAGON	69.707	28.384	41.323	3.545	3.293	2.528	3.667	8.430	14.482	13.881	19.881
PRINCIPADO DE ASTURIAS	77.938	34.663	43.275	2.849	2.593	3.020	3.572	11.652	16.083	17.142	21.027
ILLES BALEARS	53.731	22.904	30.827	3.488	3.515	2.401	3.282	7.085	11.181	9.930	12.849
PALMAS LAS	111.251	48.549	62.702	3.664	3.658	4.265	5.353	14.230	20.819	26.390	32.872
STA. CRUZ DE TENERIFE	102.517	45.804	56.713	3.183	3.065	3.724	4.779	13.715	19.942	25.182	28.927
CANARIAS	213.768	94.353	119.415	6.847	6.723	7.989	10.132	27.945	40.761	51.572	61.799
CANTABRIA	40.229	18.314	21.915	1.504	1.328	1.671	1.889	5.946	8.138	9.193	10.560
ALBACETE	37.672	14.076	23.596	1.782	1.661	1.552	2.119	4.122	7.901	6.620	11.915
CIUDAD REAL	52.990	20.094	32.896	2.816	2.874	2.299	3.410	5.973	11.734	9.006	14.878
CUENCA	14.008	5.647	8.361	641	579	556	778	1.552	2.878	2.898	4.126
GUADALAJARA	16.377	6.672	9.705	716	636	553	848	2.063	3.785	3.340	4.436
TOLEDO	67.193	25.270	41.923	3.164	3.070	2.299	3.536	7.296	15.354	12.511	19.963

Paro sexo y edad



SIGUIENTES PASOS...

Visualización de Datos

En siguientes clases utilizaremos la BD de vuelos del año 2008 de EEUU, los campos se pueden ver a continuación.

Autoguardado

Vuelos - Excel

Herramientas de tabla

Herramientas de consultas

Marin Diaz, Gabriel

MD

ArchivoInicioInsertar

Disposición de páginaFórmulasDatosRevisarVistaAyudaPower PivotSage ERP X3DiseñoConsulta

Cortar

Copiar

Copiar formato

Portapapeles

Calibri

11

A^A

N

K

Fuente

Alinear texto

Combinar y centrar

Alineación

General

Número

Formato condicional

Dar formato como tabla

Normal

Bueno

Incorrecto

Neutral

Estilos

Insertar

Eliminar

Formato

Celdas

Autosuma

Rellenar

Borrar

Ordenar y filtrar

Buscar y seleccionar

Edición

Ideas

Extract Trello

Get Jira Data

Commands Group

Jira Cloud

Ideas

Compartir

Comentarios

A1

<

Click me!

!Muchas Gracias!

GABRIEL MARÍN DÍAZ
LCDO. CIENCIAS FÍSICAS UCM

www.linkedin.com/in/gabrielmarindiaz/