Trabajando con la Nube

Sistemas Distribuidos

Grado en Ingeniería Informática

Índice

Índice

Redes sociales




Redes sociales

- Guardan gran cantidad de información.
- Permiten *agrupar* información de varias fuentes.
- Pueden ofrecer un API para acceder a la aplicación.
  - Usado por sistemas automáticos.

Autenticación

The Problem of password fatigue and identity overload

Problema de la autenticación

¿Como se puede organizar la identificación?

Sistemas de Identificación

OpenID

- Protocolo abierto y descentralizado.
- Muy usado en la web, pero algo anticuado.

Facebook

- Protocolo propio (¿espia?).
- Acceso todos los datos de Facebook.

OAuth

- Plataforma para identificar terceros.
- No envía datos, sólo autentica localmente.

Bots

¿Qué es un bot?

- Programa automático que interactúa usando la red social.
- Permite programar interacciones.

Algunos ejemplos

- Indicar que el usuario está ausente.
- Publicar en otros medios (CMS) a partir de la red social.
- Propagar notificaciones de cambios (temperatura, . . .).
- Gestionar imágenes y/o vídeos enviados por red social.
- Automatizar interacciones.

Acceso

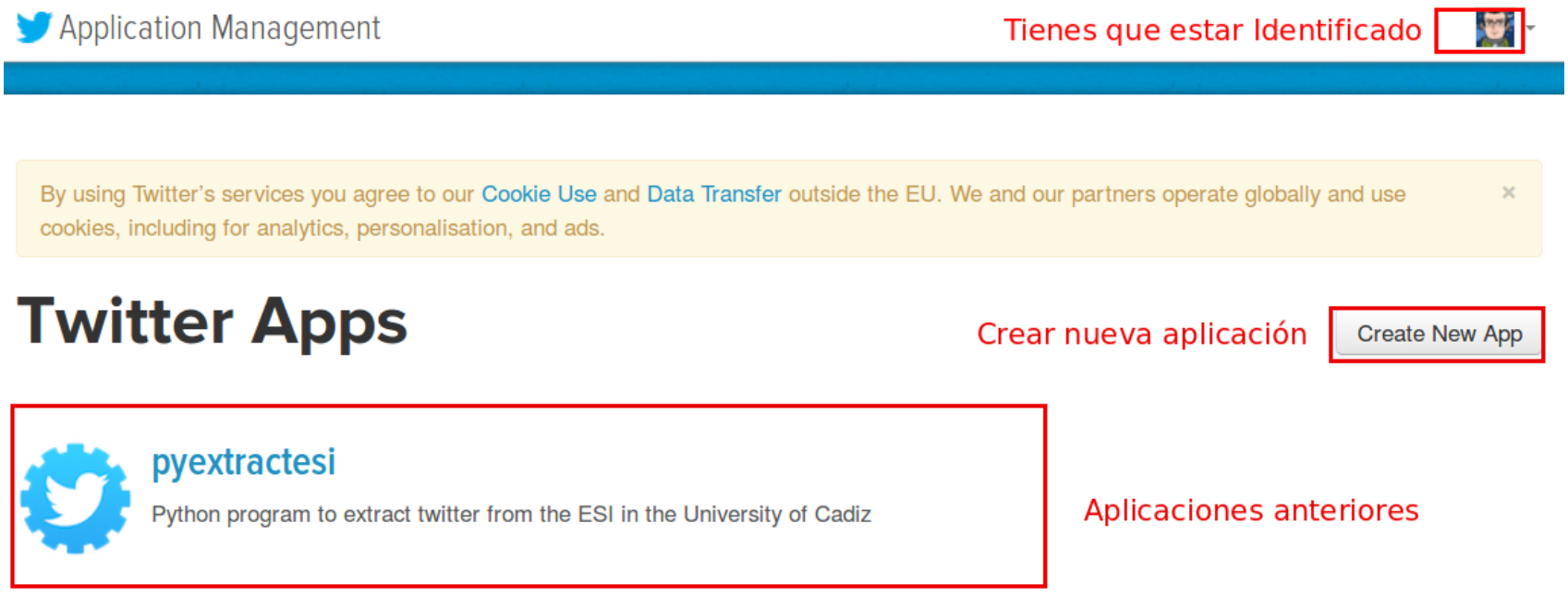
Pasos

1. Conectarse al portal de desarrollo de la red social.

   Twitter `https://apps.twitter.com/`
   Google `https://developers.google.com/`

2. Identificarse en la red social.
3. Crear una aplicación asociada a esa cuenta.
4. Dar permisos adecuados a esa aplicación.
5. Obtener certificados/claves para identificarse como la aplicación.
6. En la aplicación, usar los certificados/claves para conectarse.

Ejemplo: Twitter



Ejemplo: Twitter

Create an application

Application Details

Name *

Your application name. This is used to attribute the source of a tweet and in user-facing authorization screens. 32 characters max.

Description *

Your application description, which will be shown in user-facing authorization screens. Between 10 and 200 characters max.

Website *

Opcional en la práctica

Your application's publicly accessible home page, where users can go to download, make use of, or find out more information about your application. This fully-qualified URL is used in the source attribution for tweets created by your application and will be shown in user-facing authorization screens.
(If you don't have a URL yet, just put a placeholder here but remember to change it later.)

Callback URL

Where should we return after successfully authenticating? OAuth 1.0a applications should explicitly specify their oauth_callback URL on the request token step, regardless of the value given here. To restrict your application from using callbacks, leave this field blank.

Ejemplo: Twitter

pyejemploenclase

Details | Settings | Keys and Access Tokens | Permissions

Ejemplo de identificación en Clase

http://www.uca.es

Organization

Information about the organization or company associated with your application. This information is optional.

Organization	None
Organization website	None

Application Settings

Your application's Consumer Key and Secret are used to authenticate requests to the Twitter Platform.

Access level	Read and write (modify app permissions)
Consumer Key (API Key)	ckBJY44BHxlvs6Btq0Tbbi2C8 (manage keys and access tokens)

Ejemplo: Twitter

pyejemploenclase

Details | Settings | Keys and Access Tokens | Permissions

Ejemplo de identificación en Clase

http://www.uca.es

Organization

Information about the organization or company associated with your application. This information is optional.

Organization	None
Organization website	None

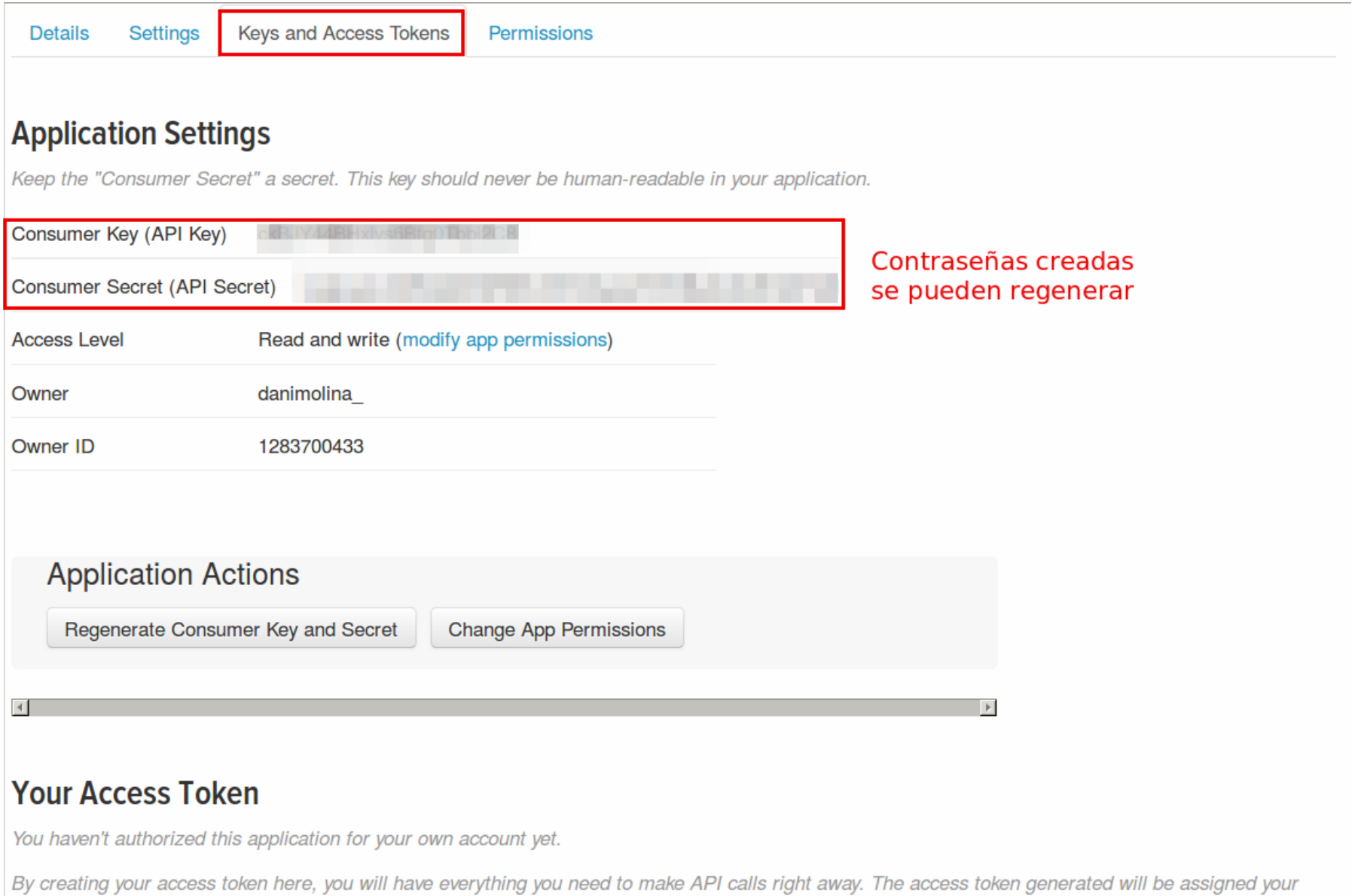
Application Settings

Your application's Consumer Key and Secret are used to authenticate requests to the Twitter Platform.

Access level	Read and write (modify app permissions)
Consumer Key (API Key)	ckBJY44BHxlvs6Btq0Tbbi2C8 (manage keys and access tokens)

Ejemplo: Twitter

Ejemplo: Twitter



Ejemplo: Twitter

Application Settings

Keep the "Consumer Secret" a secret. This key should never be human-readable in your application.

Consumer Key (API Key)

Consumer Secret (API Secret)

Access Level — Read, write, and direct messages (modify app permissions)

Owner

Owner ID

Vamos a necesitar estas 4 contraseñas

Application Actions

Regenerate Consumer Key and Secret | Change App Permissions

Your Access Token

This access token can be used to make API requests on your own account's behalf. Do not share your access token se

Access Token

Access Token Secret

Acceso a los datos en Red Social

Protocolo REST *a mano*

- Más inmediato.
- Susceptible de cambios.

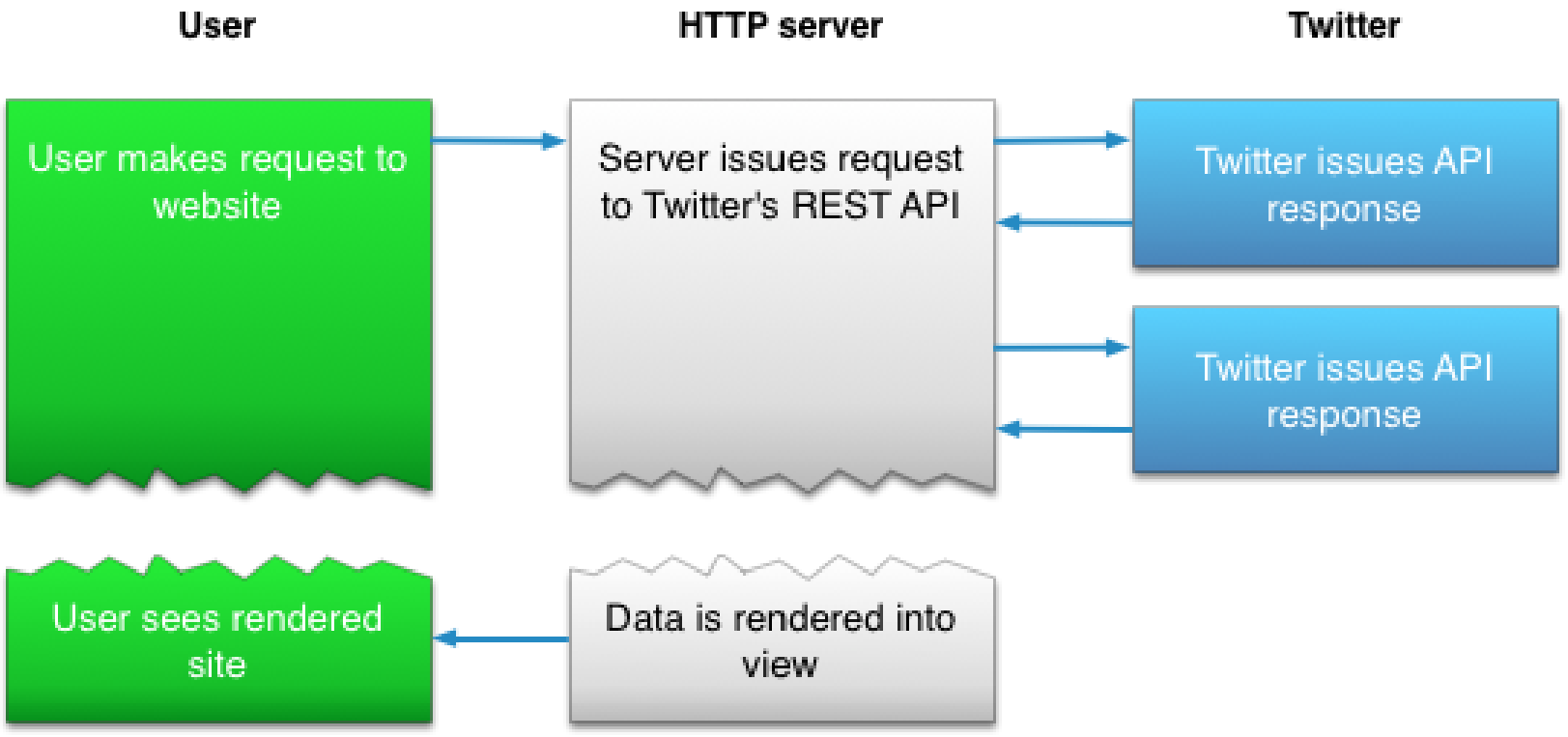
Librería de acceso

- Requiere aprender a usarlo.
- Más cercano al lenguaje.

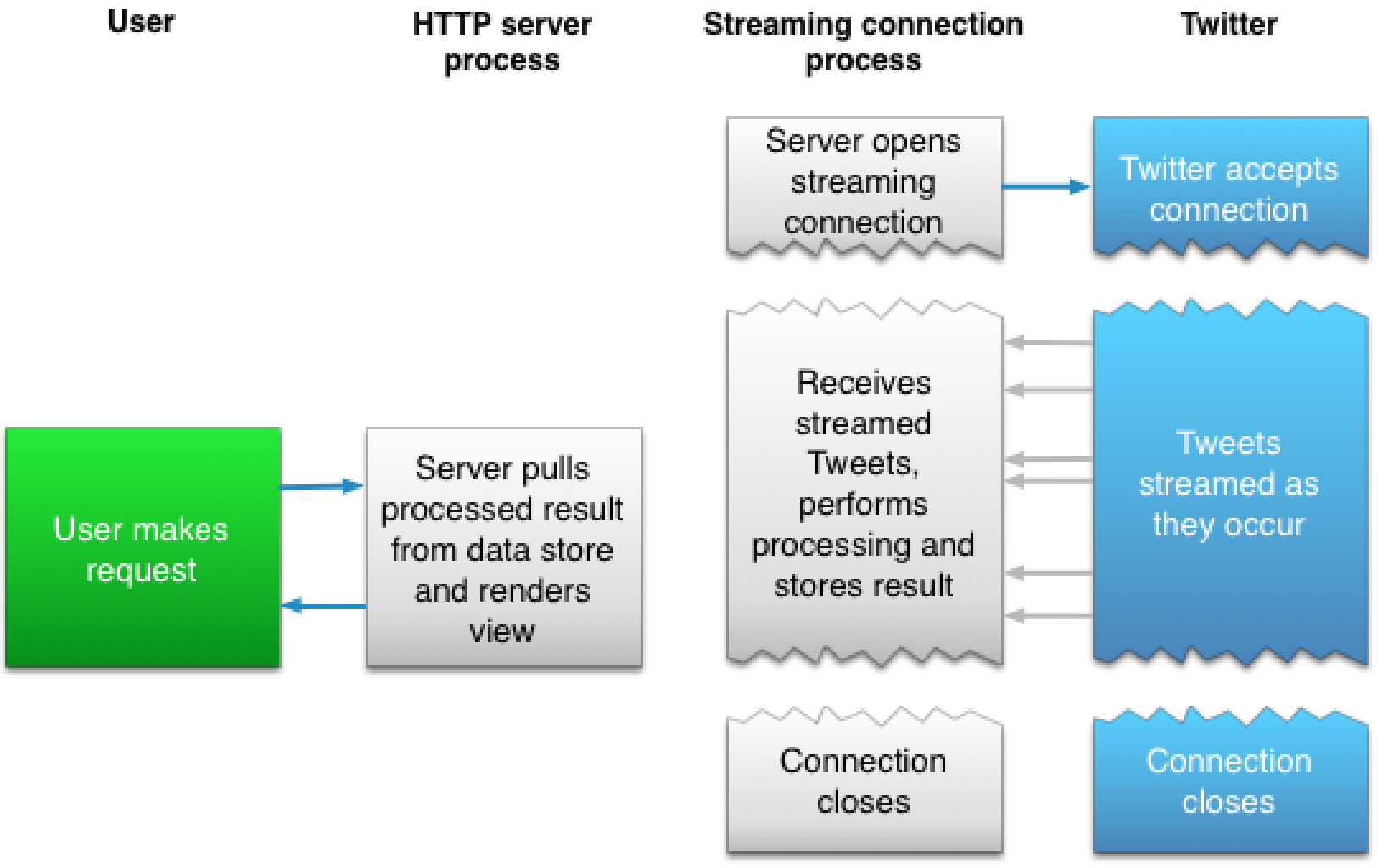
Tipo de acceso

1. Tu aplicación llama al API de la red social.
2. La red social te avisa de nuevos eventos.

Acceso mediante REST



Acceso mediante Streaming



Índice

Probando en Twitter: Librería Tweepy

Tweepy

- Librería en Python para uso de Twitter
- No sólo consulta los tweets:
  - Suscripciones.
  - *Retweets*.
- Extracción de imágenes y vídeo.
- Facilita programar *bots* en Twitter.
- Uso de Stream.

Autenticando con Tweepy

Uso de cuatro claves

```
consumer_key = "..."
consumer_secret = "..."
access_token = "..."
access_token_secret  = "..."
```

Autenticando

```
import tweepy

auth = tweepy.OAuthHandler(consumer_key, consumer_secret)
auth.set_access_token(access_token, access_token_secret)

api = tweepy.API(auth)
```

Conceptos de Tweepy

Conceptos de Tweepy

API Conexión establecida para realizar operaciones.

Status Mensaje enviado/recibido en Twitter.

Ejemplo

```
# Muestra las publicaciones del usuario o asociados
public_tweets = api.home_timeline()
# Muestra las publicaciones del propio usuario
my_tweets = api.user_timeline()
limited = [public_tweets[:10], my_twe

for status in limited:
    print(status.text)
```

Status

Status

- Permiten acceder a la información del mensaje.
- Son estructuras con muchísima información.

Información má útil

- **author** Autor del mensaje.
- **text** Texto del mensaje.
- **lang** Idioma.
- **retweet_count** Número de retuiter.
- **entities** Elementos del mensaje.
  - **hashtags** tags.
  - **url** URL indicada.
  - **media** Imágenes.

Métodos útiles de API

Listados

user_timeline() Muestra publicaciones propias.

home_timeline() Muestra publicaciones propias o ajenas.

retweets_of_me() Muestra retuiteados.

Publicar

update_status(texto, ...) Crea un nuevo mensaje.

update_with_media(filename, [texto]) Incluye una imagen.

retweet(id) Retuitea un mensaje/status.

Métodos útiles de API

Usuarios

followers() Seguidores.

search_users() Busca usuarios.

Amistad

friends_id(user_id) Muestra los que ese usuario sigue.

follower_ids(user_id) Muestra los usuarios que le siguen.

create_friendship(usuario_id) Crea *amistad* con el indicado.

destroy_friendship(usuario_id) Elimina la anterior.

exists_friendship(usuario1, usuario2) Comprueba si existe.

report_spam/create_block Bloquea al usuario indicado.

Métodos útiles de API

Búsquedas

Búsqueda search(q=query, page=page, show_user=True|False)

Se pueden guardar y recuperar

```
query_id = api.create_saved_search("conference")
...
# Recuperar los resultados con su id
list = get_saved_search(query_id)
```

Cursor

Límites por página

El cursor permite evitar el límite por página.

```
page = 1
while True:
    statuses = api.user_timeline(page=page)
    if statuses:
        for status in statuses:
            # process status here
            process_status(status)
    else:
        # All done
        break
    page += 1  # next page
```

```
for status in tweepy.Cursor(api.user_timeline).items():
    # process status here
    process_status(status)
```

Cursor

También se puede consultar sólo una página

```
cursor = tweepy.Cursor(api.user_timeline)
for page in cursor.pages():
    # page is a list of statuses
    process_page(page)
```

Paso de parámetros a función

```
  # api.user_timeline(id="twitter")
cursor = tweepy.Cursor(api.user_timeline, id="twitter")
  for status in cursor.items():
      pass
```

Streaming

Streaming

- Se puede asociar al twitter una clase.
- Por cada nuevo mensaje se llama al método on_status de la clase.

Ejemplo

```
#override tweepy.StreamListener to add logic to on_status
class MyStreamListener(tweepy.StreamListener):

    def on_status(self, status):
        print(status.text)

my_listener = MyStreamListener()
myStream = tweepy.Stream(auth = api.auth, listener=my_listener)
```

Streaming

Se puede filtrar por término

```
myStream.filter(track=["python"])
```

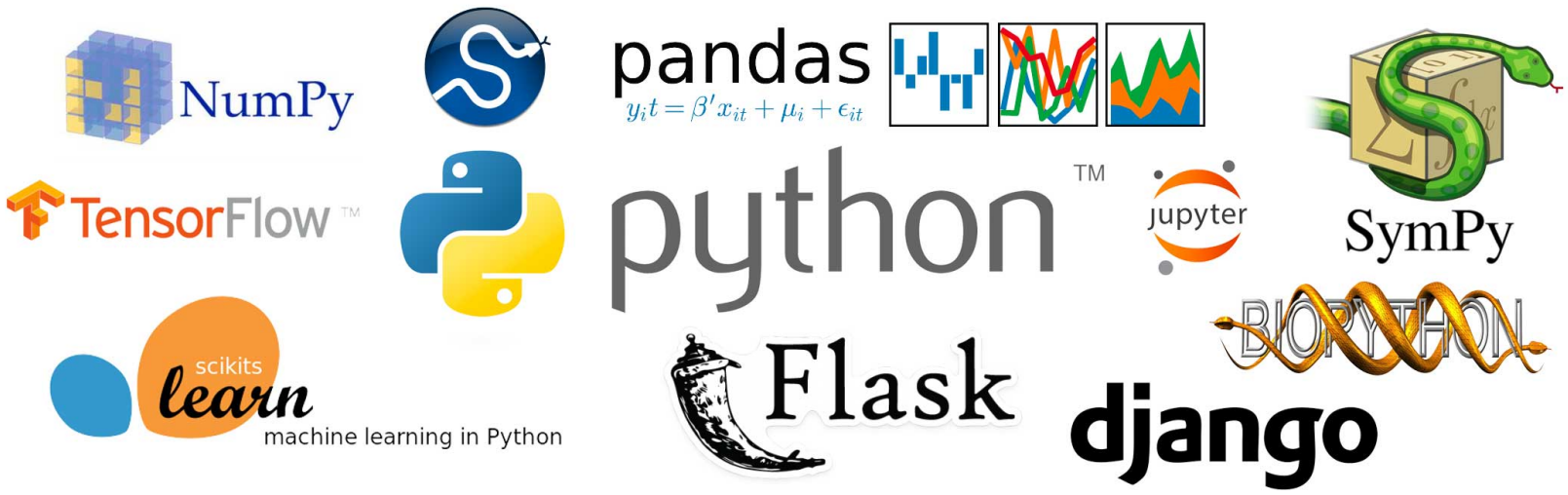
Ejemplo avanzado

```
#bot.py
import tweepy

class BotStreamer(tweepy.StreamListener):
    def on_status(self, status):
        username = status.user.screen_name
        status_id = status.id

    #entities provide structured data from Tweets including
    #resolved URLs, media, hashtags and mentions
        if "media" in status.entities:
            for image in status.entities["media"]:
                img_url = image["media_url"]
                tweet_image(img_url, username, status_id)
```

Índice

Índice

Python en cálculo científico



Python se usa mucho en cálculo científico

- Gran número de librerías para procesar datos.
- Muy usado en Ciencia de Datos.

Visión general de Librerías

Numpy

- Librería de matrices y vectores.
- Ofrece un API parecido a Matlab.
- Muy eficiente.

Pandas

- Construye el concepto de *DataFrames* sobre Python.
- Permite leer y escribir en ficheros (.csv, Excel, . . .).
- Permite visualizar los datos.

Librerías de Visualización

Matplotlib Librería, potente pero compleja.

seaborn Algo más de alto nivel.

bokeh Produce web, permite gráficas interactivas.

DataFrame

DataFrame

Almacena un dato con distintas variables/columnas.

Acceso a una columna/dimensión

Índices	df.index.
Columnas	df.columns.
Atributos/Columna	df[/columna].

Es muy flexible

```
print(df.columns)
["Name", "Surname", "Years"]
df.columns = ["Name", "Surname", "Age"]
df["Age"]
[23, 45]
df["Full name"] = df["Name"] + " " +df["Surname"]
print(df.columns)
["Name", "Surname", "Age", "Fullname"]
```

DataFrame

Crearlo

- Se puede crear a partir de un conjunto de vectores.

```
data = np.array([[,Col1,Col2],
                 [Row1,1,2],
                 [Row2,3,4]])

print(pd.DataFrame(data=data[1:,1:],
                   index=data[1:,0],
                   columns=data[0,1:]))
```

- Su uso más común es a través de ficheros .csv ó .xls.

Leyendo datos

read_csv	Lee fichero CSV.
read_excel	Lee fichero .xls (y .xlsx).

Escribiendo datos

Requiere la librería xldr no instalada con Pandas.

Acceso de datos

Columnas

	A	B	C
0	1	2	3
1	4	5	6
2	7	8	9

Acceso

Columna Por nombre

```
df["A"]
print(df["A","B"])
```

Varias columnas

Dato concreto

```
df.loc[row][col]
```

Ejemplo de pandas

Ejemplo Leyendo datos

```
import pandas as pd
from pandas import ExcelWriter
from pandas import ExcelFile

# load the sheet into the Pandas Dataframe structure
df = pd.read_excel(Pandas-Example.xlsx, sheetname=Sheet1)

print("The list of row index")
print(df.index)
print("The column headings")
print(df.columns)

print("The Patient column information:")
print(df[Patient])

# print each row of the patient column
for i in df.index:
    print(df[Patient][i])
```

Ejemplo de pandas

Escribiendo datos

```
# compute a new column as the product of two other columns
for i in df.index:
    df[BP*SO2][i] = df[BP][i] * df[SO2][i]

print("results of column multiplication:")
print(df[BP*SO2])

# write the dataframe back out with the new column data included
writer = ExcelWriter(Pandas-Example-Out.xlsx)
df.to_excel(writer,Sheet1,index=False)
writer.save()
```

Tutorial online

https://goo.gl/1eUOqx