

# BIG DATA



# Introdução ao Big Data

Tema da Aula: **Twitter REST API**

Prof.: **Dino Magri**

## **Coordenação:**

Prof. Dr. Adolpho Walter  
Pimazzi Canton

Profa. Dra. Alessandra de  
Ávila Montini

- Contatos:

- E-mail: [professor.dinomagri@gmail.com](mailto:professor.dinomagri@gmail.com)
- Twitter: [https://twitter.com/prof\\_dinomagri](https://twitter.com/prof_dinomagri)
- LinkedIn: <http://www.linkedin.com/in/dinomagri>
- Site: <http://www.dinomagri.com>

## Coordenação:

Prof. Dr. Adolpho Walter  
Pimazzi Canton

Profa. Dra. Alessandra de  
Ávila Montini

# Currículo

- **(2014-Presente)** – Professor no curso de Extensão, Pós e MBA na Fundação Instituto de Administração (FIA) – [www.fia.com.br](http://www.fia.com.br)
- **(2018-Presente)** – Pesquisa e Desenvolvimento de Big Data e Machine Learning na Beholder (<http://beholder.tech> )
- **(2013-2018)** – Pesquisa e Desenvolvimento no Laboratório de Arquitetura e Redes de Computadores (LARC) na Universidade de São Paulo – [www.larc.usp.br](http://www.larc.usp.br)
- **(2012)** – Bacharel em Ciência da Computação pela Universidade do Estado de Santa Catarina (UDESC) – [www.cct.udesc.br](http://www.cct.udesc.br)
- **(2009/2010)** – Pesquisador e Desenvolvedor no Centro de Computação Gráfica – Guimarães – Portugal – [www.ccg.pt](http://www.ccg.pt)
- **Lattes:** <http://lattes.cnpq.br/5673884504184733>

## Material das aulas

- Caso esteja utilizando seu próprio computador, realize o download de todos os arquivos e salve na **Área de Trabalho** para facilitar o acesso.
  - Lembre-se de instalar os softwares necessários conforme descrito no documento de Instalação (**InstalaçãoPython3v1.2.pdf**).
- Nos computadores da FIA os arquivos já estão disponíveis, bem como a instalação dos softwares necessários.

# Conteúdo da Aula

- Objetivo
- Introdução
- Twitter REST API
- Exercícios

# Conteúdo da Aula

- **Objetivo**
- Introdução
- Twitter REST API
- Exercícios

# Objetivo

- Objetivo dessa aula é introduzir os conceitos sobre as **APIs do Twitter** e como utilizá-las.



# Conteúdo da Aula

- Objetivo
- **Introdução**
- Twitter REST API
- Exercícios

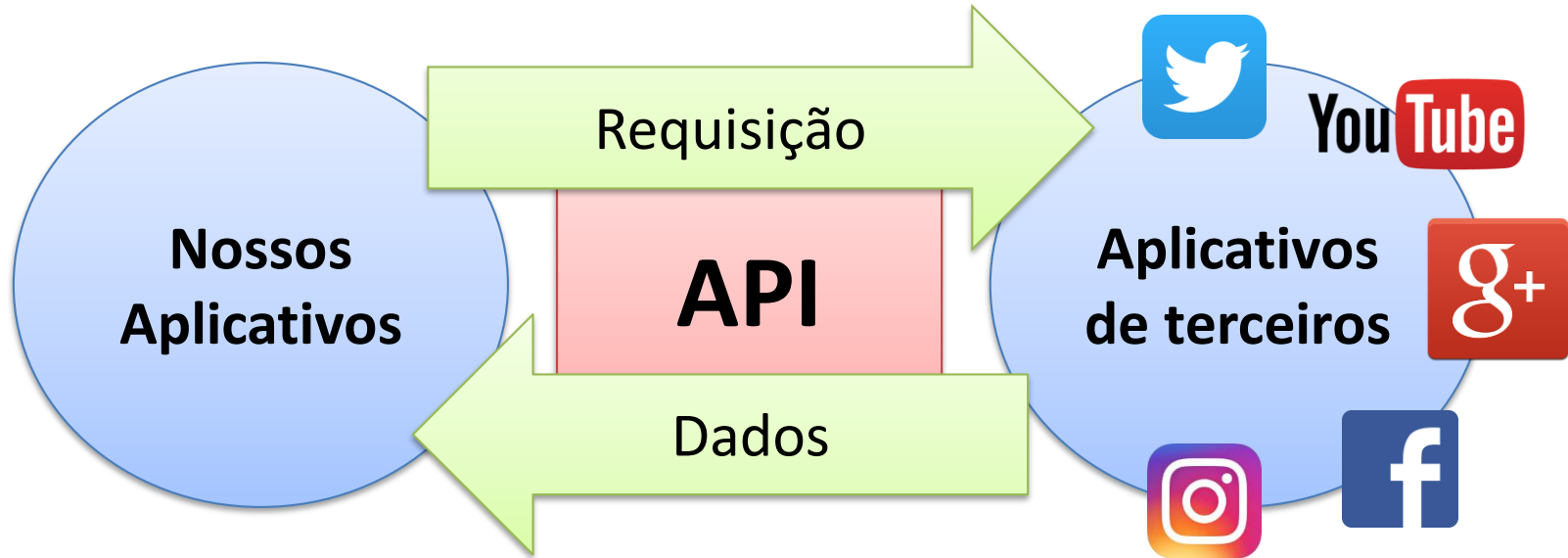
# O que é uma API?



# O que é uma API?

- É um **conjunto de padrões** para acesso a uma plataforma baseada na Web.
- É uma interface de comunicação que **possibilita que diferentes tipos de software comuniquem entre si**.
- Uma API deve fornecer:
  - Flexibilidade, Segurança, Simplicidade no Uso, Escalabilidade, Portabilidade.

# O que é uma API?



# API do Twitter

- Existem diversas APIs para acessar os dados do Twitter:

- REST API



- Streaming API



- Ads API

- Entre outras: <https://dev.twitter.com/overview/api>

# API do Twitter

- **REST API**

- Permite ler e escrever tweets via API do Twitter, além de recuperar informações sobre o perfil, seguidores, entre outros.

- **Streaming API**

- Permite ter acesso contínuo de novas respostas para as mesmas requisições das consultas realizadas pela REST API através de conexões HTTP.

- **Ads API**

- Permite integrar propaganda dos parceiros do Twitter para gerenciar seus produtos.

# API do Twitter

- Além disso, iremos utilizar **OAuth** para conectar usuários ao Twitter e enviar requisições seguras e autorizadas para as APIs do Twitter.
- Para acessar as APIs do Twitter precisamos ter acesso a desenvolvedor.
- É importante ter realizado os passos descritos na documentação: **CriarContaDesenvolvedorTwitter.pdf**

# Credenciais de acesso da API do Twitter

- Precisamos recuperar as credenciais de acesso. Para isso:
  - Acesse <https://developer.twitter.com>
  - Clique em **Sign In** do lado direito superior
  - Realize o login.
  - Acesse diretamente o endereço:  
<https://developer.twitter.com/en/apps>
    - Ou clique no nome de usuário que aparece e depois em Apps.



# Credenciais de acesso da API do Twitter

- Clique em **Keys and tokens!**
- Pronto! Já temos as credenciais de acesso:
  - Consumer Key (API Key)
  - Consumer Secret (API Secret)
  - Access Token
  - Access Token Secret

## Módulo Tweepy

- Para acessar a API do Twitter iremos utilizar um módulo do Python chamado Tweepy.

# Tweepy

An easy-to-use Python library for accessing the Twitter API.

- Permite atualizar o status
- Permite receber informações do perfil
- Permite monitorar os tweets e realizar alguma ação quando um evento acontecer.
- Permite receber os tweets públicos.
- Seguir todos os usuários que seguirem o usuário autenticado.
- E muito mais!

<http://docs.tweepy.org/en/latest/api.html>

# Módulo Tweepy

- Para instalar o módulo Tweepy, abra o CMD (terminal para Linux e MacOS X) e digite:
  - `pip install tweepy`
    - `oauthlib` – biblioteca que permite o uso do protocolo de autorização segura.
    - `requests` – permite o uso de envio de requisições HTTP.
    - `requests-oauthlib` – possibilita o suporte da biblioteca OAuth para o requests.
    - `six` – mantém a compatibilidade entre Python 2 e 3.

# Módulo Tweepy

- Para testar é só importar o módulo:

```
>>> import tweepy
```

# Conteúdo da Aula

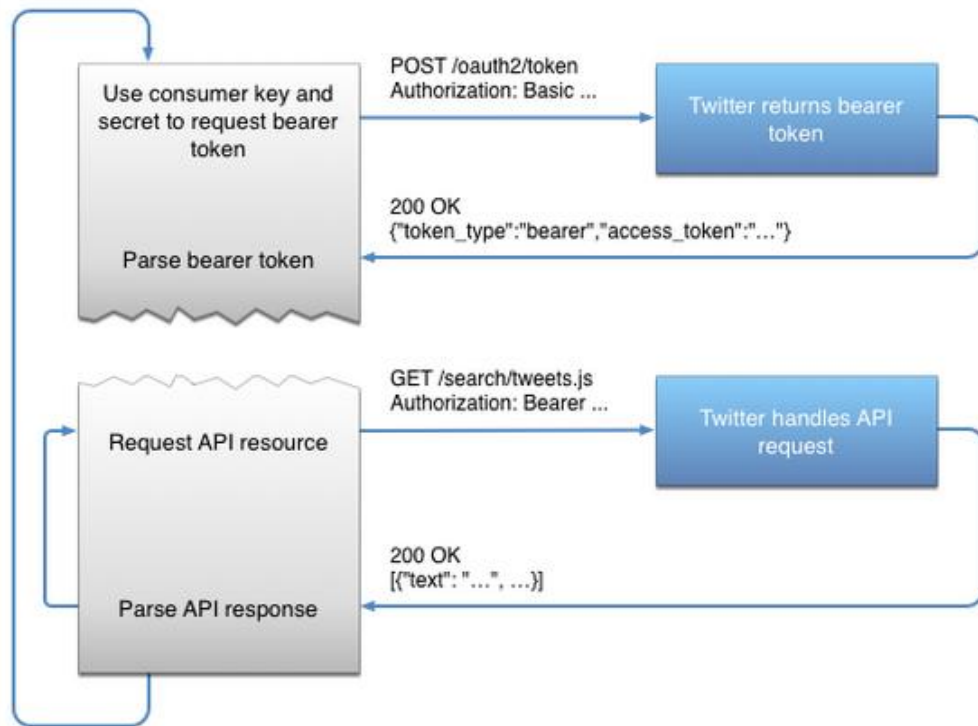
- Objetivo
- Introdução
- **Twitter REST API**
- Exercícios

# Twitter REST API

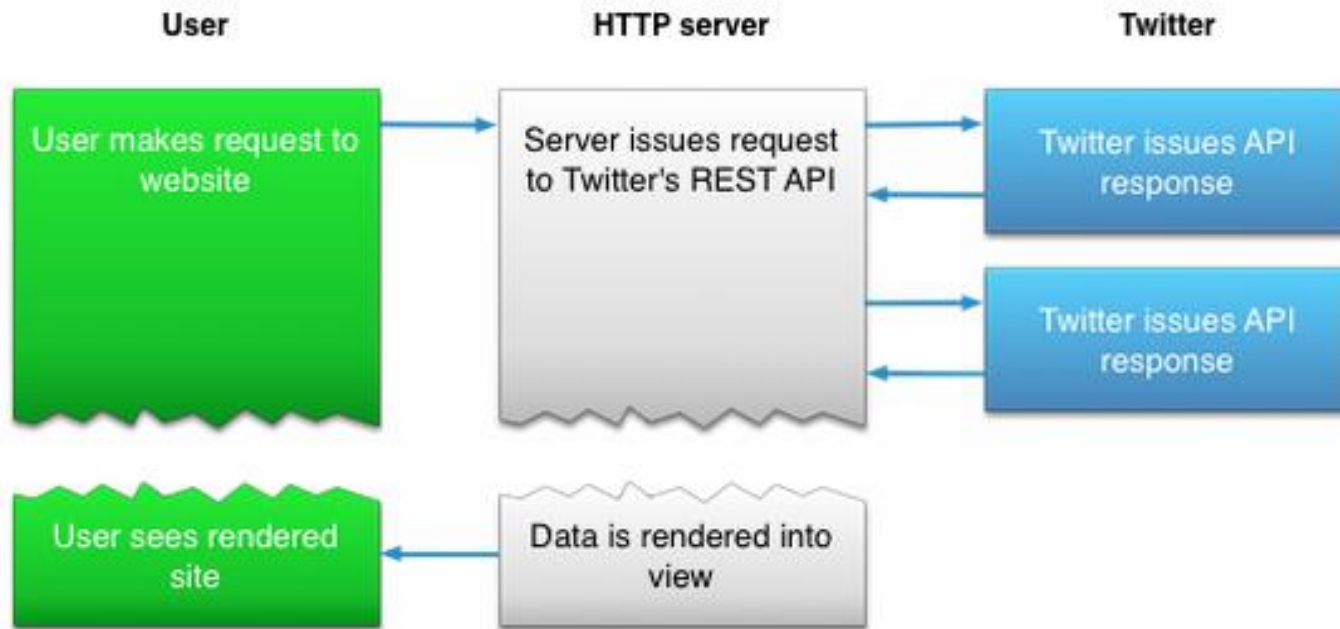
- Para utilizar a API do Twitter, temos que garantir que as requisições sejam seguras e autorizadas, para isso iremos utilizar o **OAuth**.



Fonte: <https://dev.twitter.com/oauth/application-only>



# Twitter REST API



Fonte: <https://dev.twitter.com/rest/public>



# Twitter REST API

- Iremos criar programas para:
  - Publicar um tweet na timeline autenticada
  - Recuperar dados
  - Pesquisar tweets que tenham determinados termos

# Algumas considerações

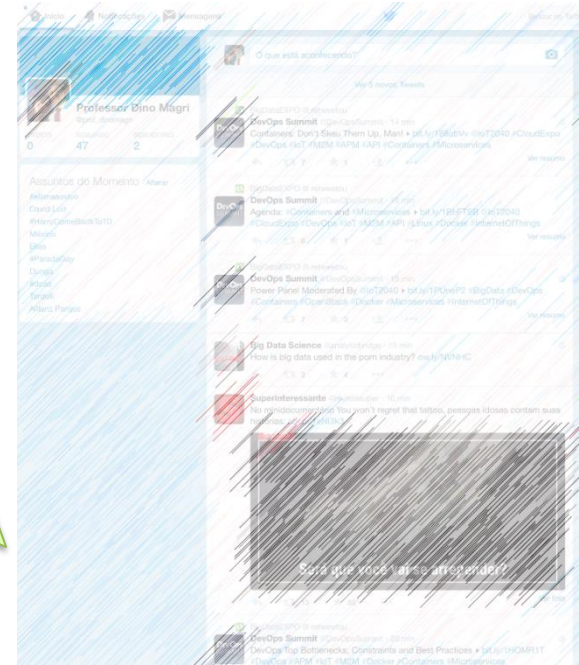
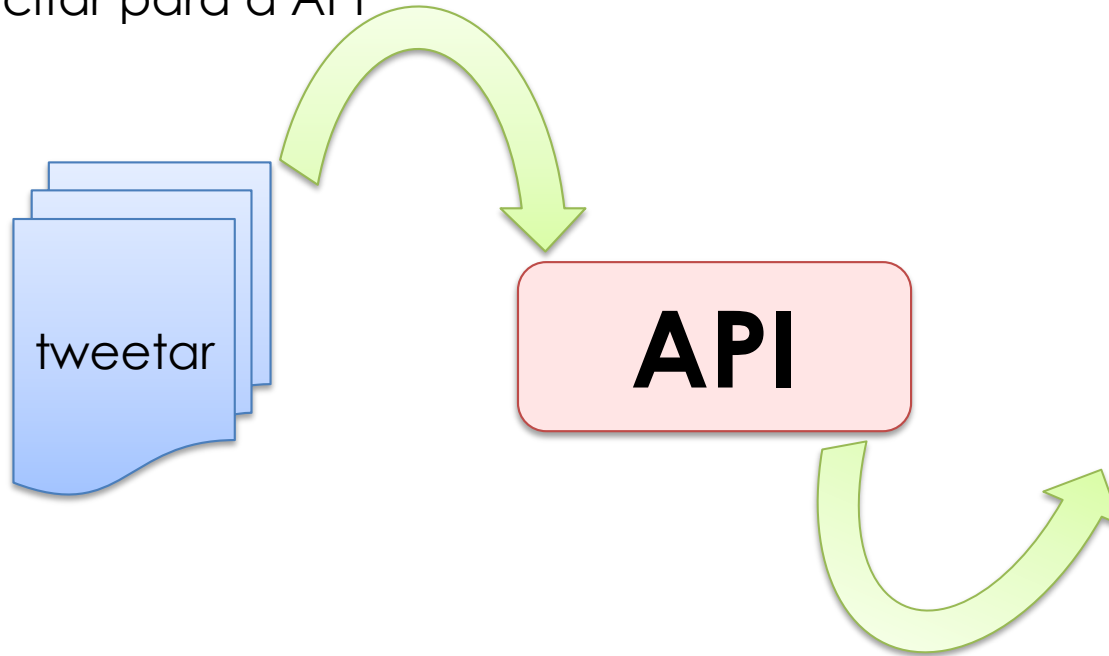
- Limites da API
  - GET:
    - Search: 180 (user) ou 450 (app) chamadas a cada 15 minutos
    - <https://developer.twitter.com/en/docs/basics/rate-limiting>
    - <https://developer.twitter.com/en/docs/basics/rate-limits.html>
  - POST: <https://support.twitter.com/articles/289867>
    - Mensagens diretas: 1000/dia
    - Tweets: 2400/dia
    - Seguir: 1000/dia

# Chega de papo! Vamos criar códigos!



# Publicar um Tweet

Solicitar para a API



Publicar um tweet na timeline autenticada

# Publicar um Tweet

- Primeiro passo do nosso código é importar o módulo tweepy e criar strings que devem conter as chaves e tokens de acesso.

```
import tweepy  
  
consumer_key = ''  
  
consumer_secret = ''  
  
access_token = ''  
  
access_token_secret = ''
```

## Publicar um Tweet

- Através da classe **OAuthHandler** do módulo tweepy iremos criar autorização e depois definiremos o acesso.

```
autorizar = tweepy.OAuthHandler(consumer_key,  
consumer_secret)
```

```
autorizar.set_access_token(access_token,  
access_token_secret)
```

# Publicar um Tweet

- Atualmente o Twitter **não define prazo de validade dos tokens**, assim, é necessário reinvocar o direito de acesso da aplicação para finalizar o seu uso.
- Uma vez autorizado, queremos ter acesso aos métodos disponíveis na API.

```
api = tweepy.API(autorizar)
```

- Tuitando 😊

```
api.update_status(status="Big Data")
```

Fonte: <https://dev.twitter.com/rest/reference/post/statuses/update>

# Publicar um Tweet

- O retorno será:

```
Status(in_reply_to_status_id=None, favorite_count=0, truncated=False,
id_str='758354516060299264', contributors=None,
in_reply_to_screen_name=None, lang='pt', created_at=datetime.datetime(2016,
7, 27, 17, 33), in_reply_to_user_id=None, source='fia-python',
author=User(geo_enabled=True, friends_count=53, screen_name='dinomagri',
id=3311191877, profile_use_background_image=True, verified=False,
created_at=datetime.datetime(2015, 6, 6, 22, 16, 52), is_translator=False,
profile_background_image_url='http://abs.twimg.com/images/themes/theme1/bg.p
ng', ... continua ...
```



# Publicar um Tweet

- Quando invocamos um método da API (e.g. `update_status`), o Twitter **realiza** a ação e **retorna** a resposta em formato JSON.
- O Tweepy por sua vez realiza a instancia da classe **Models**.
- Desta forma, poderemos acessar os dados que foram retornados e utilizar dentro de nossa aplicação. Por exemplo:

```
>>> retorno = api.update_status(status="O que é big data?")  
>>> print(type(retorno))  
<class 'tweepy.models.Status'>
```

# Publicar um Tweet

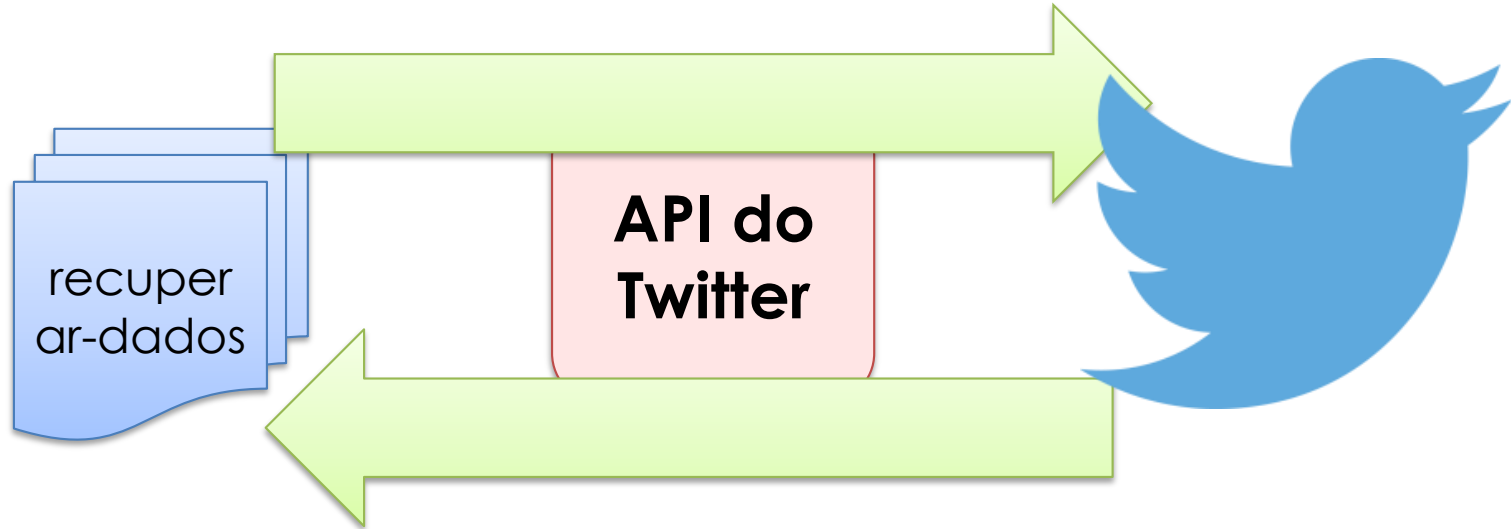
```
>>> print(dir(retorno))  
['author', 'contributors', 'coordinates', 'created_at', 'destroy', 'entities',  
'favorite', 'favorite_count', 'favorited', 'geo', 'id', 'id_str',  
'in_reply_to_screen_name', 'in_reply_to_status_id', 'in_reply_to_status_id_str',  
'in_reply_to_user_id', 'in_reply_to_user_id_str', 'is_quote_status', 'lang',  
'parse', 'parse_list', 'place', 'retweet', 'retweet_count', 'retweeted',  
'retweets', 'source', 'source_url', 'text', 'truncated', 'user']  
  
>>> print(retorno.id)  
758361343275704320  
  
>>> api.destroy_status(id=retorno.id)  
  
Status(in_reply_to_status_id=None, favorite_count=0, truncated=False,  
id_str='758361343275704320' ... continua ...
```



Abra o arquivo "**aula4-parte1-atualizando-status.ipynb**"

# Recuperar Tweets

Solicitar para a API os *tweets* públicos da *timeline*



Twitter irá devolver os dados solicitados

# Recuperar Tweets

- Para criar esse programa, temos que:
  - Importar o módulo
  - Definir as chaves e tokens de acesso
  - Criar a autorização
  - Definir o acesso
  - Criar a API com base no acesso

# Recuperar Tweets

```
import tweepy

consumer_key = ''

consumer_secret = ''

access_token = ''

access_token_secret = ''

autorizar = tweepy.OAuthHandler(consumer_key, consumer_secret)

autorizar.set_access_token(access_token, access_token_secret)

api = tweepy.api(autorizar)
```

# Recuperar Tweets

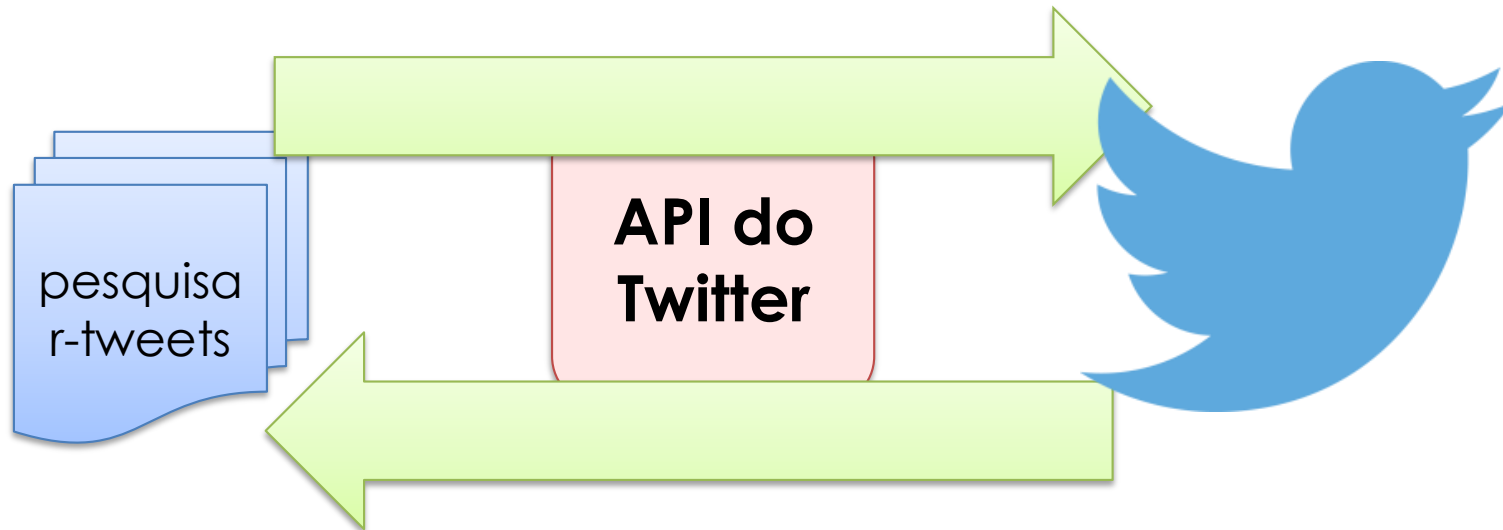
- Vamos utilizar a `home_timeline()` para recuperar as últimas 20 atualizações, incluindo retweets, da timeline do usuário autenticado.
- Vamos utilizar a `user_timeline()` para recuperar as 5 atualizações de um usuário específico.
- Vamos utilizar a `retweets_of_me()` para recuperar os retweets que foram feitos dos meus posts.



Abra o arquivo "**aula4-parte2-recuperar-tweets.ipynb**"

# Pesquisar por Tweets

Solicitar para a API pesquisar por *tweets* públicos que combinam com um termino definido



Twitter irá devolver os dados solicitados

# Pesquisar por Tweets

- Na API REST podemos utilizar o método **search** para procurar por tweets que combinam com o termo definido.
- O método contém algumas opções como:

```
api.search(q, count, max_id, lang)
```

- **q** é o parâmetro que terá o termo a ser pesquisado.
- **count** é a quantidade de tweets que serão retornados. O limite é 100 e o padrão é 15.
- **max\_id** retorna apenas os tweets com o ID menor ou igual ao que foi especificado.
- **lang** restringe a busca por tweets de um determinado idioma.



## Pesquisar por Tweets

- Para criar esse programa, temos que realizar os mesmos passos iniciais que fizemos nos programas anteriores.
  - Importar o módulo, definir as chaves e tokens de acesso, criar a autorização, definir o acesso e criar a API.

## Pesquisar por Tweets

- Por padrão, o método `search` permite retornar apenas os 100 primeiros tweets.
- Mesmo que o parâmetro `count` seja definido, por exemplo, `count=1000`.
- Para recuperar 1000 tweets, precisamos criar algumas estruturas de controle para que seja possível salvar esses tweets.

# Pesquisar por Tweets

Como podemos fazer isso?

## Pesquisar por Tweets

- Primeiro, vamos criar uma lista para salvar os tweets, definir uma variável para pegar o último id que foi recuperado e outra para a quantidade de tweets que queremos buscar.

```
tweets_salvos = []
```


```
ultimo_id = -1
```

```
qtde_tweets = 1000
```

## Pesquisar por Tweets

- Agora precisamos criar um **laço de repetição** que irá repetir enquanto não recuperarmos os 1000 elementos que serão salvos na lista `tweets_salvos`.

**Não execute esse código ainda!!!!**



```
while len(tweets_salvos) < qtde_tweets:  
    contador = qtde_tweets - len(tweets_salvos)
```

# Pesquisar por Tweets

- Como estamos trabalhando com a API do Twitter e iremos criar um laço até preencher os 1000 elementos, temos que **cuidar de possíveis erros**. Por exemplo:
  - Pode acontecer da internet cair e a conexão com o Twitter seja finalizada.
  - Pode acontecer de não existir os 1000 elementos.
  - Pode acontecer do Twitter impor algum limite de uso.
  - Entre outras situações.
- Python facilita bastante a forma para tratamento de exceções.



Abra o arquivo **"aula4-parte3-tratamento-excecoes.ipynb"**

## Pesquisar por Tweets

- Agora que sabemos como tratar erros em nosso código, podemos tentar **recuperar os tweets e caso aconteça algum erro, iremos imprimir e finalizar o laço de repetição.**
- O método `api.search` tem o parâmetro `max_id`, que permite recuperar os tweets anteriores ao valor definido no `max_id`.
- Com isso, podemos recuperar os tweets do último até o 100º, do 101º até 200º, 201º até 300º e assim sucessivamente.

# Pesquisar por Tweets

- Com o par **try/except**, vamos utilizar a `api.search` passando a palavra (`q`) que queremos buscar, o contador (`count`) e vamos definir o `max_id` para recuperar os tweets anteriores ao último id recuperado.

```
try:
    novos_tweets = api.search(q='Python',
count=contador, max_id=str(ultimo_id - 1))
except tweepy.TweepError as e:
    print("Erro:", (e))
    break
```



## Pesquisar por Tweets

- Se nenhum tweet for recuperado, temos que finalizar o laço. Podemos fazer uma verificação na lista `novos_tweets`, **se estiver vazia iremos finalizar o laço** (`break`).
- Caso contrário, iremos estender a lista `tweets_salvos` com os dados da lista `novos_tweets`, utilizando o método `extend`.
- Também **precisamos atualizar** o `ultimo_id` para que no próximo laço retorne os 100 anteriores ao `ultimo_id`.

# Pesquisar por Tweets

**try:**

```
    novos_tweets = api.search(q='Python',
count=contador, max_id=str(ultimo_id - 1))
    if not novos_tweets:
        print("Nenhum tweet para recuperar")
        break
    tweets_salvos.extend(novos_tweets)
    ultimo_id = novos_tweets[-1].id
except tweepy.TweepError as e:
    print("Erro:", (e))
    break
```

## Pesquisar por Tweets

- Com isso, podemos lidar com possíveis erros e também salvar os 1000 tweets que queremos.
- Por fim, vamos imprimir todos os tweets.

```
for i, tweet in enumerate(tweets_salvos, start=1):  
    print("{} ---- {}".format(i, tweet.text))
```

```

tweets_salvos = []
ultimo_id = -1
qtde_tweets = 1000
while len(tweets_salvos) < qtde_tweets:
    contador = qtde_tweets - len(tweets_salvos)
    try:
        novos_tweets = api.search(q='Python', count=contador,
max_id=str(ultimo_id - 1))
        if not novos_tweets:
            print("Nenhum tweet para recuperar")
            break
        tweets_salvos.extend(novos_tweets)
        ultimo_id = novos_tweets[-1].id
    except tweepy.TweepError as e:
        print("Erro:", (e))
        break
for i, tweet in enumerate(tweets_salvos, start=1):
    print("{} ---- {}".format(i, tweet.text))

```



Abra o arquivo "aula4-parte4-pesquisar-tweets.ipynb"

# Conteúdo da Aula

- Objetivo
- Introdução
- Twitter REST API
- **Exercícios**

 Abra o arquivo "**aula4-parte5-exercicios.ipynb**"

# Exercícios

- **Exercício 1** – Utilizando o método `update_with_media`, realize a atualização do status utilizando a imagem `fia.jpg` disponível na pasta da aula.
  - Além da imagem, adicione a mensagem: "Programação com Python e Twitter na FIA!"
  - Lembre-se de utilizar a função `help(api.update_with_media)` para ver os nomes dos parâmetros necessários.

## Exercícios

- **Exercício 2** - Salve o retorno do tweet do exercício anterior e imprima as seguintes informações:

- **tweet**

- `id, created_at, lang, text`

- **user**

- `screen_name, friends_count, time_zone`

Por fim, remova o tweet, utilizando o método `destroy_status`.

## Exercícios

- **Exercício 3** - Utilizando o método `home_timeline()`, recupere os 10 tweets atuais. Para cada um desses tweets, imprima:
  - o `screen_name`
  - o texto do tweet
  - o id do usuário



## Exercícios

- **Exercício 4** - Para cada tweet do exercício anterior, utilize o id do usuário e recupere os 5 primeiros tweets de cada um dos 10 usuários (`user_timeline`).



Abra o arquivo "**aula4-parte5-exercícios-gabarito.ipynb**"

## Referências Bibliográficas

- **21 Recipes for Mining Twitter** – Matthew A. Russell – USA: O'Reilly, 2011.
- **Mastering pandas** – Femi Anthony – Packt Publishing, 2015.
- **Python for Data Analysis** – Wes McKinney – USA: O'Reilly, 2013.
- Referência da API do Tweepy -  
<http://docs.tweepy.org/en/latest/api.html>

## Referências Bibliográficas

- **Python for kids – A playful Introduction to programming** – Jason R. Briggs – San Francisco – CA: No Starch Press, 2013.
- **Python Cookbook** – David Beazley & Brian K. Jones – O'Reilly, 3th Edition, 2013.