

Aula 12 - APIS

A sigla API deriva da expressão inglesa *Application Programming Interface* que, traduzida para o português, pode ser compreendida como uma interface de programação de aplicação. Ou seja, API é um conjunto de normas que possibilita a comunicação entre plataformas através de uma série de padrões e protocolos.

Dessa forma, existem muitas APIs criadas para fornecer dados a outros programas. De forma geral mandamos uma requisição de alguma forma para a API e ela nos retorna algo (podemos também mandar requisições para realizar alterações, etc...porém não é o escopo desta aula.)

Uma requisição pode ser feita simplesmente por meio de um URL. Considere a API do OSRM (opensource routing machine), que fornece informações geográficas por meio de sua API. Por meio da seguinte URL:

```
http://router.project-osrm.org/route/v1/driving/13.388860,52.517037;13.397634,52.529407;13.428555,52.523219?overview=false
```

Conseguimos acessar as distâncias entre as 3 localizações (longitude,latitude) contidas na URL. A resposta vem em um formato chamado *JSON* (isso depende da API e dos parâmetros passados). Com esse formato, conseguimos extrair os dados necessário.

Podemos visualizar os elementos de um arquivo JSON em um visualizador online para simplificação. Considere a resposta da API acima, insira a mesma no seguinte site (<http://jsonviewer.stack.hu/>), e veja como a informações está estruturada.

12.1 - Fazendo requisições de API em Python

Para coeltarmos a mesma informação acima direto do nosso programa em Python, precisamos usar a biblioteca `requests`, junto a função `get`. Abaixo segue um exemplo de uso em que são coletados dados da API (<https://economia.awesomeapi.com.br/last/USD-BRL,EUR-BRL,BTC-BRL>), que retorna cotações atualizadas de moedas.

```
In [1]: import requests as rq

# awesome api
```

```
respostas = rq.get("https://economia.awesomeapi.com.br/last/USD-BRL,EUR-BRL,BTC-BRL")
```

```
# As respostas estão em um json que está dentro de respostas  
respostas.json()['USDBRL']
```

```
Out[1]: {'code': 'USD',  
        'codein': 'BRL',  
        'name': 'Dólar Americano/Real Brasileiro',  
        'high': '5.1167',  
        'low': '5.1166',  
        'varBid': '0.0004',  
        'pctChange': '0.01',  
        'bid': '5.1162',  
        'ask': '5.1171',  
        'timestamp': '1675119247',  
        'create_date': '2023-01-30 19:54:07'}
```

12.2 - Exemplos de APIs em Python

API que fornece a hora atualizada (<http://worldtimeapi.org>):

Horário mundial

```
In [10]: # World time API  
respostas = rq.get("http://worldtimeapi.org/api/timezone/America/Sao_Paulo")  
respostas.json()["datetime"]
```

```
Out[10]: '2022-09-14T17:39:18.426733-03:00'
```

Geoprocessamento/ matriz de distancias

API que fornece a matriz de distância entre pontos geográficos (<http://router.project-osrm.org>):

```
In [34]: # OSRM  
respostas = rq.get('http://router.project-osrm.org/table/v1/driving/-49.26742667248318,-25.43800114521021;-49.2834
```

```
M = respostas.json()["distances"]
M
```

Out[34]: [[0, 3146.4], [2059.4, 0]]

Mapas do gogle maps

API do google (precisa de uma chave), que busca um mapa estático.

```
In [3]: local = "Curitiba,Brazil"
chave = "XXXXXXXXXXXXXXXXXXXXXXXXX"
respostas = rq.get("https://maps.googleapis.com/maps/api/staticmap?center=" + local + "&zoom=14&size=400x400&key="

# Abre um arquivo para salvar o conteúdo
f = open(r'G:\Meu Drive\Arquivos\UFPR\Disciplinas\2 - Intro Mineração de Dados\file.jpeg', 'wb')
f.write(respostas.content)
f.close()

# Abre a imagem
from IPython import display
display.Image(r'G:\Meu Drive\Arquivos\UFPR\Disciplinas\2 - Intro Mineração de Dados\file.jpeg')
```

Out[3]:



Dados do governo federal (propostas da camara dos deputados deferais)

API do governo que fornece dados da camara dos deputados (<https://dadosabertos.camara.leg.br>).

```
In [11]: # https://api.sic.org.economia.gov.br/  
respostas = rq.get("http://estruturaorganizational.dados.gov.br/doc/estrutura-organizational/resumida?codigoPoder=  
respostas.json()["servico"]
```

```
Out[11]: {'codigoErro': 0,  
          'mensagem': 'Processamento sem erros',  
          'data': '2022-09-17',  
          'versaoServico': '3.10.0',  
          'ipRequisitante': '191.245.173.207',  
          'ticket': None}
```

```
In [10]: # https://dadosabertos.camara.leg.br/#
respostas = rq.get("https://dadosabertos.camara.leg.br/api/v2/eventos?ordem=ASC&ordenarPor=dataHoraInicio", header
respostas.json()["dados"][0]
```

```
Out[10]: {'id': 66366,
'uri': 'https://dadosabertos.camara.leg.br/api/v2/eventos/66366',
'dataHoraInicio': '2022-09-14T14:00',
'dataHoraFim': None,
'situacao': 'Convocada',
'descricaoTipo': 'Audiência Pública',
'descricao': 'Vacinação das pessoas com diabetes e com obesidade\r\n REUNIÃO CONJUNTA',
'localExterno': None,
'orgaos': [{'id': 2014,
'uri': 'https://dadosabertos.camara.leg.br/api/v2/orgaos/2014',
'sigla': 'CSSF',
'nome': 'Comissão de Seguridade Social e Família',
'apelido': 'Seguridade Social e Família',
'codTipoOrgao': 2,
'tipoOrgao': 'Comissão Permanente',
'nomePublicacao': 'Comissão de Seguridade Social e Família',
'nomeResumido': 'Seguridade Social'},
{'id': 537871,
'uri': 'https://dadosabertos.camara.leg.br/api/v2/orgaos/537871',
'sigla': 'CIDOSO',
'nome': 'Comissão de Defesa dos Direitos da Pessoa Idosa',
'apelido': 'Defesa dos Direitos da Pessoa Idosa',
'codTipoOrgao': 2,
'tipoOrgao': 'Comissão Permanente',
'nomePublicacao': 'Comissão de Defesa dos Direitos da Pessoa Idosa',
'nomeResumido': 'Idosos'}],
'localCamara': {'nome': 'Anexo II, Plenário 12',
'predio': None,
'sala': None,
'andar': None},
'urlRegistro': 'https://www.youtube.com/watch?v=DqgH2lRw4hE'}
```

Mercado de ações

Para coletar dados de ações podemos usar o próprio pacote do pandas `datareader`.

```
In [4]: import pandas as pd
from pandas_datareader import data as web

# Bovespa
df = web.DataReader(f'^BVSP', data_source='yahoo', start=f'02-20-2020', end='02-20-2021')
print(df)

# Facebook
df2 = web.DataReader("FB", data_source='yahoo', start=f'08-20-2020', end='12-20-2020')
print(df2)

# Alcoa corporation
df3 = web.DataReader("AA", data_source='yahoo', start=f'08-20-2020', end='12-20-2020')
print(df3)
```

Date	High	Low	Open	Close	Volume	Adj Close
2020-02-20	116552	114379	116518	114586	6793000	114586
2020-02-21	114585	112661	114585	113681	5896200	113681
2020-02-26	113647	105053	113647	105718	9369800	105718
2020-02-27	106656	102984	105711	102984	11237700	102984
2020-02-28	104172	99951	102984	104172	11228400	104172
...
2021-02-11	120283	118440	118440	119235	7567400	119235
2021-02-12	119763	118163	119300	119116	7655900	119116
2021-02-17	120573	118880	119421	120391	8326100	120391
2021-02-18	120845	118515	120361	119140	10720600	119140
2021-02-19	119250	117867	119199	118748	10402800	118748

[245 rows x 6 columns]

Date	High	Low	Open	Close	Volume	\
2021-12-07	326.540009	321.000000	321.570007	322.809998	18794000	
2021-12-08	332.750000	323.070007	325.000000	330.559998	19937700	
2021-12-09	336.130005	328.000000	329.540009	329.820007	16879200	
2021-12-10	335.029999	326.369995	332.559998	329.750000	14528000	
2021-12-13	341.089996	329.589996	330.950012	334.489990	22948700	
...	
2022-12-01	121.199997	118.400002	119.199997	120.440002	36551400	
2022-12-02	124.040001	117.610001	117.830002	123.489998	39875100	
2022-12-05	124.669998	121.349998	121.750000	122.430000	35474900	
2022-12-06	120.550003	113.739998	119.910004	114.120003	43645300	
2022-12-07	115.879997	112.879997	113.760002	113.430000	15469568	

Date	Adj Close
2021-12-07	322.809998
2021-12-08	330.559998
2021-12-09	329.820007
2021-12-10	329.750000
2021-12-13	334.489990
...	...
2022-12-01	120.440002
2022-12-02	123.489998
2022-12-05	122.430000
2022-12-06	114.120003

2022-12-07 113.430000

[253 rows x 6 columns]

	High	Low	Open	Close	Volume	Adj Close
Date						
2020-08-20	15.245000	14.730000	15.040000	15.19	3140900	15.045506
2020-08-21	15.040000	14.515000	14.910000	14.60	4326700	14.461119
2020-08-24	15.130000	14.700000	14.850000	15.12	3171200	14.976171
2020-08-25	15.250000	14.820000	15.120000	15.05	2363300	14.906838
2020-08-26	15.440000	15.020000	15.090000	15.07	3158500	14.926647
...
2020-12-14	23.375000	21.681999	23.250000	21.76	6286200	21.553009
2020-12-15	22.600000	21.757999	22.115000	22.17	4226500	21.959108
2020-12-16	22.129999	21.304001	22.059999	21.58	5423800	21.374723
2020-12-17	23.209999	21.740000	22.280001	22.18	7318400	21.969011
2020-12-18	22.610001	21.860001	22.209999	22.01	5137200	21.800631

[85 rows x 6 columns]

Dados do banco central

Podemos acessar series temporais do banco central (<https://www3.bcb.gov.br/sgspub/localizarseries/localizarSeries.do?method=prepararTelaLocalizarSeries>). Para acessarmos a API referente a esses dados:

(<https://dadosabertos.bcb.gov.br/dataset/20542-saldo-da-carteira-de-credito-com-recursos-livres---total/resource/6e2b0c97-afab-4790-b8aa-b9542923cf88>)

In [18]: `import pandas as pd`

```
# https://api.bcb.gov.br/dados/serie/bcdata.sgs.{codigo_serie}/dados/ultimos/{N}?formato=json
url = "https://api.bcb.gov.br/dados/serie/bcdata.sgs.432/dados/ultimos/500?formato=json" #SELIC META
df = pd.read_json(url)
df["valor"].max()
df["valor"].min()
```

Out[18]: 5.25