

CURSO

Python aplicado a finanzas



MÓDULO 3

Data y conectores

Retomamos **Diccionarios** y sumamos **JSON**

DICCIONARIOS

- Es un tipo de dato muy utilizado.
- Tiene operaciones propias.
- Puedo guardar cualquier otro tipo de dato.
- Se crea con dos llaves vacías {}
- Maneja un par de elementos clave: valor.

¿QUÉ ES JSON?

- JSON (JavaScript Object Notation): es un formato de texto sencillo para el intercambio de datos.
- Se trata de un subconjunto de la notación literal de objetos de JavaScript.
- Debido a su amplia adopción como alternativa a XML, se considera un formato independiente del lenguaje

```
[ ] json = {  
    "atributo1": "valor1",  
    "atributo2": "valor2"  
}
```

¿Qué es JSON?

- XML: eXtensible Markup Language = “Lenguaje de Marcado Extensible” o “Lenguaje de Marcas Extensible”.
- Permite definir lenguajes de marcas Utilizado para almacenar datos en forma legible.
- Proviene del lenguaje SGML y permite definir la gramática de lenguajes específicos.
- A diferencia de otros lenguajes, XML da soporte a bases de datos, siendo útil cuando varias aplicaciones deben comunicarse entre sí o integrar información.

```
[ ] ""  
    <xml>  
        <atributo1>  
            valor1  
        </atributo1>  
        <atributo2>  
            valor2  
        </atributo2>  
    </xml>  
    ""
```



¿Cómo proveemos de datos al programa?

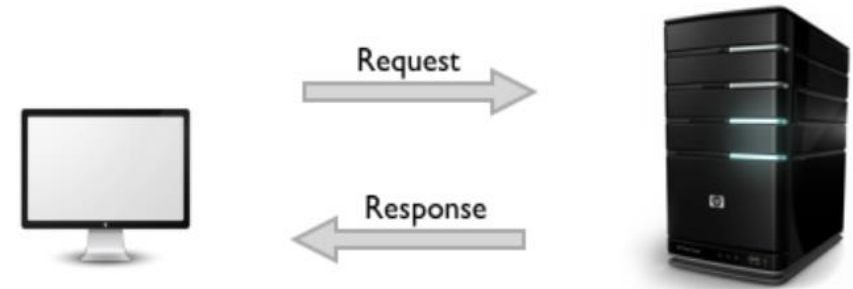
- Los programas se alimentan de los datos provistos.
- Como se provisiona depende de como se los haya pensado a dicho programa.
- Los diversos modos de aprovisionamientos pueden ser (entre otros):
 - Por medio ingreso de quienes usan al programa
 - Por medio de información en archivos.
 - Por medio de intercambio en la red.
 - Otros.



Arquitectura **Cliente - Servidor**



- Es un modelo de diseño de software en el que las tareas se reparten entre los proveedores de recursos o servicios, llamados servidores, y los demandantes, llamados clientes.
- Un cliente realiza peticiones a otro programa, el servidor, quien le da respuesta.



Arquitectura cliente servidor

Protocolo **HTTP/1**

- Protocolo de transferencia de hipertexto (en inglés, Hypertext Transfer Protocol, abreviado HTTP)
- Permite las transferencias de información en la World Wide Web.
- **HTTP** define la sintaxis y la semántica que utilizan los elementos de software de la arquitectura web (clientes, servidores, proxies) para comunicarse.



- HTTP es un protocolo sin estado, es decir, no guarda ninguna información sobre conexiones anteriores.
- El desarrollo de aplicaciones web necesita frecuentemente mantener estado.
- Para esto se usan las cookies, que es información que un servidor puede almacenar en el sistema cliente

URL

- Un **HTTP URL** combina en una dirección simple los cuatro elementos básicos de información necesarios para recuperar un recurso desde cualquier parte en Internet:
 - ✓ El protocolo que se usa para comunicar o enviar datos.
 - ✓ El anfitrión (servidor o host) con el que se comunica.
 - ✓ El puerto de red en el servidor para conectarse.
 - ✓ La ruta al recurso en el servidor (por ejemplo, su nombre de archivo).

- Un **URL típico** puede ser del tipo:

<http://es.wikipedia.org:80/wiki/Special:Search?search=tren&go=Go>

- Dónde:
 - ✓ **http** es el protocolo.
 - ✓ **es.wikipedia.org** es el anfitrión (host).
 - ✓ **80** es el número de puerto de red en el servidor.
 - ✓ **/wiki/Special:Search** es la ruta de recurso (path).
 - ✓ **?search=tren&go=Go** es la cadena de búsqueda (Parte opcional) query params.

Métodos de petición

- HTTP define una serie de métodos de request (algunas veces referido como "verbos") que pueden utilizarse.
- El protocolo tiene flexibilidad para ir añadiendo nuevos métodos y para así añadir nuevas funcionalidades.
- Cada método indica la acción que desea que se efectúe sobre el recurso identificado.
- Lo que este recurso representa depende de la aplicación del servidor

GET

- El método GET solicita una representación del recurso especificado.
- Las solicitudes que usan GET solo deben recuperar datos y no deben tener ningún otro efecto.

HEAD

- Pide una respuesta idéntica a la que correspondería a una petición GET, pero en la respuesta no se devuelve el cuerpo
- Esto es útil para poder recuperar los metadatos de los encabezados de respuesta, sin tener que transportar todo el contenido.

POST

- Envía datos para que sean procesados por el recurso identificado en la URL de la línea petición
- Los datos se incluirán en el cuerpo de la petición.

PUT

- Envía datos al servidor, pero a diferencia del método POST la URI de la línea de petición no hace referencia al recurso que los procesará, sino que identifica a los propios datos
- Otra diferencia con POST es semántica (ver REST): mientras que POST está orientado a la creación de nuevos contenidos, PUT está más orientado a la actualización de los mismos

Partes del **Request**

RUTA

VERBO

PARÁMETROS

BODY

HEADERS

Response:

A cada request le corresponde un response, este devuelve un objeto con información, entre ellos el código de estado, el mensaje y encabezados (headers)

- **1xx: Respuestas informativas.** Indica que la petición ha sido recibida y se está procesando.
- **2xx: Respuestas correctas.** Indica que la petición ha sido procesada correctamente.
- **3xx: Respuestas de redirección.** Indica que el cliente necesita realizar más acciones para finalizar la petición.
- **4xx: Errores causados por el cliente.** Indica que ha habido un error en el procesamiento de la petición a causa de que el cliente ha hecho algo mal.
- **5xx: Errores causados por el servidor.** Indica que ha habido un error en el procesamiento de la petición a causa de un fallo en el servidor.



Intercambio y/u obtención de Información



Web Scraping

Extraer información de la web de modo automatico a traves de analizar un documento de la web.

Ventajas: no tengo limitaciones de rate limit como si puedo tener en las apis

Desventaja: Estructura variable de los documentos, pueden banear la ip.

De una sola página web (spider):

- Procedimiento
- Obtengo URL
- Realizo una petición
- Obtengo y analizo la respuesta
- Me quedo con la información que necesito de la respuesta
- Busco otra url y repito el procedimiento

De varias páginas webs (crawls spider):

- Crawling Vertical (Acceder a cada página de cada ítem, en búsqueda de por ej el detalle del producto)
- Crawling horizontal (visitar diferentes páginas para buscar más información)

```
self.file = None
self.fingerprints
self.logdupes = True
self.debug = debug
self.logger = logger
if path:
    self.file =
    self.file.se
    self.fingerp
```

```
@classmethod
def from_settings(c
debug = setting
return cls(job,
```

```
def request_seen(s
fp = self.requ
if fp in self
```



#Ejemplo de web scraping

```
import pandas as pd
```

```
sp500 = pd.read_html('https://en.wikipedia.org/wiki/List_of_S%26P_500_companies')
```

```
sp500 = sp500[0]
```

```
[ ] pd.read_html('https://en.wikipedia.org/wiki/List_of_S%26P_500_companies')
```

	Symbol	Security	...	CIK	Founded
0	MMM	3M Company	...	66740	1902
1	ABT	Abbott Laboratories	...	1800	1888
2	ABBV	AbbVie Inc.	...	1551152	2013 (1888)
3	ABMD	ABIOMED Inc	...	815094	1981
4	ACN	Accenture plc	...	1467373	1989
..
500	YUM	Yum! Brands Inc	...	1041061	1997
501	ZBRA	Zebra Technologies	...	877212	1969
502	ZBH	Zimmer Biomet	...	1136869	1927
503	ZION	Zions Bancorp	...	109380	1873
504	ZTS	Zoetis	...	1555280	1952

[505 rows x 9 columns],

	Date	...	Reason
	Date	...	Reason
0	October 12, 2020	...	Chevron acquired Noble Energy.[6]
1	October 9, 2020	...	S&P 500 constituent Fortive spun off Vontier.[6]
2	October 7, 2020	...	Morgan Stanley acquired E*TRADE.[7]
3	September 21, 2020	...	Market capitalization change.[8]
4	September 21, 2020	...	Market capitalization change.[8]
..
254	December 5, 2000	...	Market Cap changes.
255	December 5, 2000	...	Market Cap changes.
256	December 5, 2000	...	Market Cap changes.
257	July 27, 2000	...	Market Cap change.[206]
258	December 7, 1999	...	Market Cap change.[207]

[259 rows x 6 columns]]



Módulo Requests, Get y Response

- https://www.w3schools.com/python/ref_requests_response.asp

```
[ ] import requests

url = 'http://www.google.com/search?start=0&num=10&q=rofex'
response = requests.get(url)

print(response.status_code == 200)
```



¿Qué son las APIs?

- Es una abreviatura de **Application Programming Interfaces**, que en español significa interfaz de programación de aplicaciones.
- Conjunto de definiciones y protocolos que se utiliza para desarrollar e integrar el software de las aplicaciones, permitiendo la comunicación entre dos aplicaciones de software a través de un conjunto de reglas.

API REST

- ✓ En el campo de las APIs, el acrónimo REST significa (Representational State Transfer-Transferencia de Estado Representacional).
- ✓ El formato más usado en la actualidad es el formato JSON, ya que es más ligero y legible en comparación al formato XML.
- ✓ REST se apoya en HTTP, los verbos que utiliza son exactamente los mismos, con ellos se puede hacer GET, POST, PUT, DELETE y patch.

CARACTERÍSTICAS

- ✓ Públicas
- ✓ Privadas
- ✓ Gratuitas
- ✓ Pagas
- ✓ Pueden tener seguridad y requerir un api Token



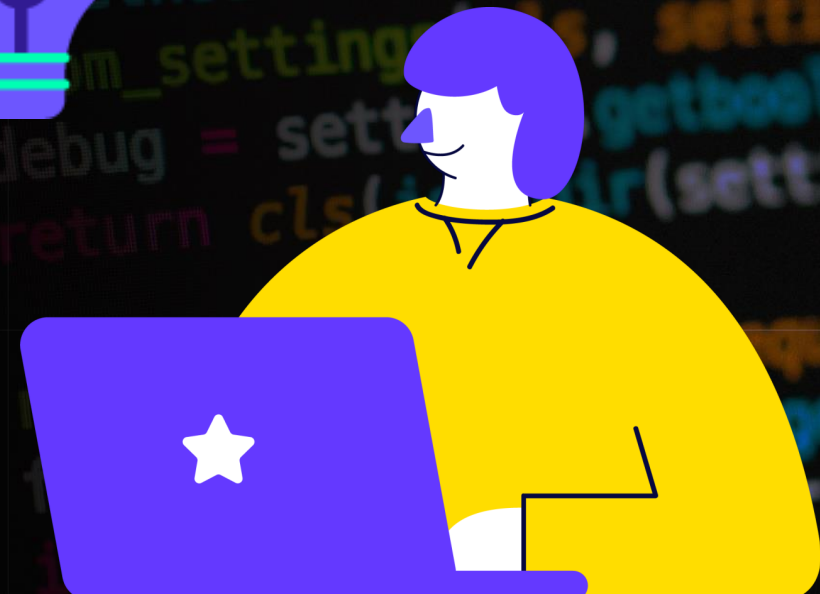
¿Qué un Token y API restful?



- ✓ La **tokenización**, cuando se aplica a la seguridad de los datos, es el proceso de sustitución de un elemento de datos sensible por un equivalente no sensible, denominado token, que no tiene un significado o valor extrínseco o explotable.
- ✓ El **token** es una referencia (es decir, un identificador) que regresa a los datos sensibles a través de un sistema de tokenización.
- ✓ El mapeo de datos originales a un token utiliza métodos que hacen que los tokens no sean factibles de revertir en ausencia del sistema de tokenización, por ejemplo, utilizando tokens creados a partir de números aleatorios.
- ✓ Por ello una de las nuevas tendencias en cuanto al desarrollo web moderno se refiere, es la autenticación por medio de Tokens y que nuestro backend sea un **API RESTful** sin información de estado, stateless.
- ✓ El funcionamiento es el siguiente. El usuario se autentica en nuestra aplicación, bien con un par usuario/contraseña, o a través de un proveedor como puede ser Twitter, Facebook o Google por ejemplo. A partir de entonces, cada petición HTTP que haga el usuario va acompañada de un Token en la cabecera. Este Token no es más que una firma cifrada que permite a nuestro API identificar al usuario. Pero este Token no se almacena en el servidor, si no en el lado del cliente (por ejemplo en localStorage o sessionStorage) y el API es el que se encarga de descifrar ese Token y redirigir el flujo de la aplicación en un sentido u otro.
- ✓ Como los tokens son almacenados en el lado del cliente, no hay información de estado y la aplicación se vuelve totalmente escalable. Podemos usar el mismo **API** para diferentes aplicaciones (Web, Mobile, Android, iOS, ...) solo debemos preocuparnos de enviar los datos en formato JSON y generar y descifrar tokens en la autenticación y posteriores peticiones HTTP a través de un middleware.

Qué es **Pypi** y **Pip**

- **Pip** es un sistema de gestión de paquetes utilizado para instalar y administrar paquetes de software escritos en Python.
- Muchos paquetes pueden ser encontrados en el **Python Package Index (PyPI)**
- **pip** es un acrónimo recursivo que se puede interpretar como **Pip Instalador de Paquetes** o **Pip Instalador de Python**.



Módulo Requests

- **Requests** es una librería HTTP de Python.
- El objetivo del proyecto es hacer que las solicitudes HTTP sean más simples y amigables para las personas.
- **Requests** es una de las bibliotecas de Python más populares que no se incluye con Python, se ha propuesto que las solicitudes se distribuyan con Python de forma predeterminada.

```
[ ] #Ejemplo en reqres
import requests
BASE_URL = "https://reqres.in/"
id = 2

response = requests.get(f"{BASE_URL}api/users/{id}").json()
response

{'data': {'avatar': 'https://reqres.in/img/faces/2-image.jpg',
'email': 'janet.weaver@reqres.in',
'first_name': 'Janet',
'id': 2,
'last_name': 'Weaver'},
'support': {'text': 'To keep ReqRes free, contributions towards server costs are appreciated!',
'url': 'https://reqres.in/#support-heading'}}
```

Ejemplo con PUT

```
[ ] import requests

body = {
    "name": "morpheus",
    "job": "zion resident"
}

url = "http://reqres.in/api/users/2"

requests.put(url, data= body).json()

{'updatedAt': '2021-11-07T23:47:26.831Z'}
```

Ejemplo con DELETE

```
[ ] import requests

url = "http://reqres.in/api/users/2"

requests.delete(url).text
```


API de alpaca

- registro: <https://alpaca.markets/>
- documentación:

```
[ ] import requests

#Client ID:
api_key = "PKQAB8SHQTA7DQGBB441"

#Client Secret:
secret_key = "xIk1NPP6kjSdgGeeUdtgzdearNuhlZUrST5eJdjc"

def getAccountInfo():

    headers = {"APCA-API-KEY-ID" : api_key, "APCA-API-SECRET-KEY":secret_key}
    endpoint = "https://paper-api.alpaca.markets/v2/account"

    r = requests.get(url = endpoint, headers = headers)

    js = r.json()
    return js
```



Market Data

TimeFrame:

- 1Min
- 5Min
- 15Min
- 1D (day)

	t	o	h	l	c	v
0	2020-12-02 05:00:00	8.77	9.2700	8.6901	9.11	561710
1	2020-12-03 05:00:00	9.10	9.2800	8.9401	8.97	462399
2	2020-12-04 05:00:00	9.02	9.1800	8.9450	9.06	544265
3	2020-12-07 05:00:00	9.07	9.1100	8.8727	8.96	456522
4	2020-12-08 05:00:00	9.00	9.0600	8.6100	8.72	367391
...
95	2021-04-21 04:00:00	7.18	7.2455	7.0600	7.23	426565
96	2021-04-22 04:00:00	7.22	7.3400	7.1700	7.18	240471
97	2021-04-23 04:00:00	7.19	7.2800	7.1000	7.21	427338
98	2021-04-26 04:00:00	7.20	7.4400	7.2000	7.41	667231
99	2021-04-27 04:00:00	7.40	7.8100	7.4000	7.71	890514

100 rows × 6 columns

```
[ ] import pandas as pd

def getHistory(symbol, timeframe='1D'):

    h = {"APCA-API-KEY-ID" : api_key, "APCA-API-SECRET-KEY":secret_key}
    par = {'symbols' : symbol}
    endpoint = "https://data.alpaca.markets/v1/bars/"+timeframe

    r = requests.get(url = endpoint, headers =h, params=par)

    df = r.json()
    df = pd.DataFrame(df[symbol])
    df.t = pd.to_datetime(df.t, unit='s')
    return df

getHistory("GGAL")
```

Conectándonos a IOL

```
[ ] """
    https://api.invertironline.com/token
    POST /token HTTP/1.1
    Host: api.invertironline.com
    Content-Type: application/x-www-form-urlencoded
    username=MIUSUARIO&password=MICONTRASEÑA&grant_type=password

    """
    import requests

    h = {
        "Content-Type": "application/x-www-form-urlencoded"
    }
    body = {
        "username": USER,
        "password": PASS,
        "grant_type": "password"
    }

    BASE_URL = "https://api.invertironline.com"

    response = requests.post(BASE_URL + "/token", headers= h, data= body).json()

    token = response["access_token"]
    token
```



```
[ ] def obtener_token():

    import requests

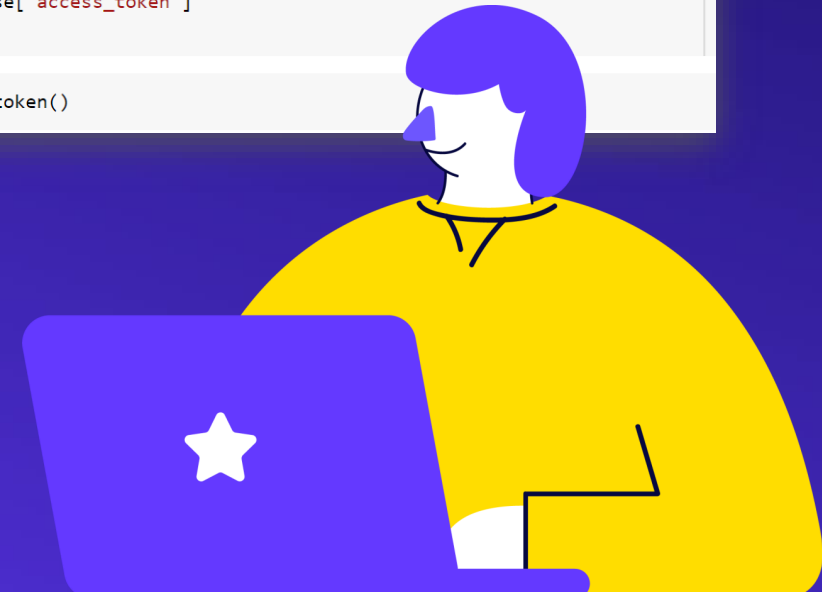
    h = {
        "Content-Type": "application/x-www-form-urlencoded"
    }
    body = {
        "username": USER,
        "password": PASS,
        "grant_type": "password"
    }

    BASE_URL = "https://api.invertironline.com"

    response = requests.post(BASE_URL + "/token", headers= h, data= body).json()

    token = response["access_token"]
    return token

[ ] token = obtener_token()
```



Market Data

```
[ ] import requests

mercado = "bCBA"
simbolo = "GGAL"
fechaDesde = "2018-06-05"
fechaHasta = "2019-02-25"
ajustada = "sinAjustar"
path = f"/api/v2/{mercado}/Titulos/{simbolo}/Cotizacion/seriehistorica/{fechaDesde}/{fechaHasta}/{ajustada}"
api_token = obtener_token()
h = {
    "Authorization": "Bearer " + api_token
}

data = requests.get(BASE_URL + path, headers = h).json()
```

```
[ ] import pandas as pd

pd.DataFrame(data)
```

Fondos Comunes de Inversión

```
[ ] import requests
```



```
endpoint = BASE_URL + "/api/v2/Titulos/FCI"
```

```
data = requests.get(endpoint, headers = headers ).json()
```

```
[ ] import pandas as pd
```

```
df = pd.DataFrame(data)
```

```
df.to_excel("fcis.xlsx")
```


Obtenemos los instrumentos por país y luego los paneles



```
[ ] import requests

pais = "argentina"
endpoint = BASE_URL + f"/api/v2/{pais}/Titulos/Cotizacion/Instrumentos"
requests.get(endpoint, headers= headers ).json()
```

```
[{'instrumento': 'Acciones', 'pais': 'argentina'},
 {'instrumento': 'Bonos', 'pais': 'argentina'},
 {'instrumento': 'Opciones', 'pais': 'argentina'},
 {'instrumento': 'Cauciones', 'pais': 'argentina'},
 {'instrumento': 'Futuros', 'pais': 'argentina'},
 {'instrumento': 'FCI', 'pais': 'argentina'}]
```

```
[ ] import requests

instrumento = 'Acciones'
endpoint = BASE_URL + f"/api/v2/{pais}/Titulos/Cotizacion/Paneles/{instrumento}"

requests.get(endpoint, headers = headers).json()
```

```
[{'panel': 'Merval'},
 {'panel': 'Panel General'},
 {'panel': 'Merval 25'},
 {'panel': 'Merval Argentina'},
 {'panel': 'Burcap'},
 {'panel': 'CEDEARs'}]
```

Operaciones de **mi cuenta**

Estado de cuenta

```
[ ] import requests
```

```
    path = '/api/v2/estadocuenta'
```

```
    endpoint = BASE_URL + path
```

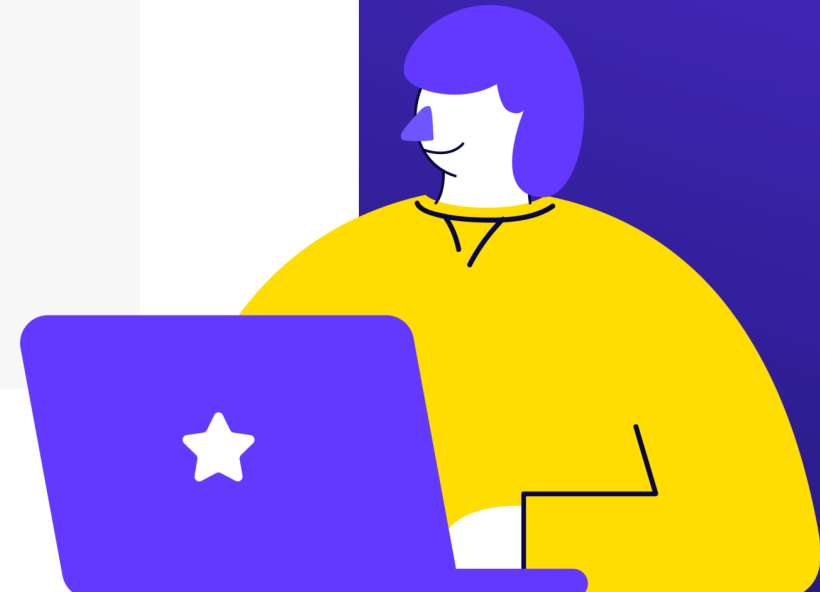
```
    data = requests.get(endpoint, headers = headers).json()
```



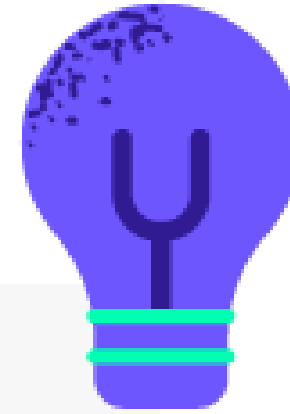
Portafolio

```
[ ] pais= "argentina"  
    #pais= "estados_Unidos"  
    path = f'/api/v2/portafolio/{pais}'  
    endpoint = BASE_URL + path
```

```
data = requests.get(endpoint, headers = headers).json()
```



Operaciones Vigentes



```
[ ] import requests
    endpoint = BASE_URL + '/api/v2/operaciones'

    data = requests.get( endpoint, headers = headers).json()
    df = pd.DataFrame(data)
    df
```

Ruteo de Órdenes



```
[ ] #previo a solicitar la compra/venta averiguamos el precio del activo
    ticker = "COME"
    mercado = "bCBA"
    endpoint = f"{BASE_URL}/api/v2/{mercado}/Titulos/{ticker}/Cotizacion"

    data = requests.get(headers = headers, url= endpoint).json()
```

```
[ ] cantidad_venta = data["puntas"][0]['cantidadVenta']
    precio_venta = data["puntas"][0]['precioVenta']
```


Compra



```
[ ] import datetime as dt
    vigencia = dt.datetime.now() + dt.timedelta(days=1)
    vigencia_str = dt.datetime.strftime(vigencia, '%Y-%m-%d')

    ticker = "COME"
    cantidad = 1
    precio = 3
    plazo = "t2"
    params = {
        "mercado": "bCBA",
        "simbolo": ticker,
        "cantidad": cantidad,
        "precio": precio,
        "plazo": plazo,
        "validez": vigencia_str
    }

    endpoint = BASE_URL + '/api/v2/operar/Comprar/'

    data = requests.post(endpoint, headers = headers, data = params).json()
    data


{'numeroOperacion': 39050848}
```

Cancelar una orden

```
[ ] data["numeroOperacion"]
```

```
39050848
```

```
[ ] nro_operacion = data["numeroOperacion"]  
requests.get(BASE_URL+f"/api/v2/operaciones/{nro_operacion}", headers = headers).json()
```



```
{  
  'aranceles': [],  
  'arancelesARS': 0.0,  
  'arancelesUSD': 0.0,  
  'cantidad': 1.0,  
  'estadoActual': 'cancelada',  
  'estados': [{  
    'detalle': 'Iniciada', 'fecha': '2021-11-08T16:06:31.913'},  
    {  
      'detalle': 'En Proceso', 'fecha': '2021-11-08T16:06:32.49'},  
    {  
      'detalle': 'Pendiente de Cancelación', 'fecha': '2021-11-08T16:08:19.313'},  
    {  
      'detalle': 'Cancelada', 'fecha': '2021-11-08T16:08:22.113'}  
  ]},  
  'fechaAlta': '2021-11-08T16:06:31.913',  
  'fechaOperado': None,  
  'fondosParaOperacion': None,  
  'mercado': 'bcba',  
  'modalidad': 'precio_Limite',  
  'moneda': 'peso_Argentino',  
  'monto': 3.0,  
  'montoOperacion': 0.0,  
  'numero': 39050848,  
  'operaciones': [],  
  'precio': 3.0,  
  'simbolo': 'COME',  
  'tipo': 'compra',  
  'validez': '2021-11-09T00:00:00'}
```

