

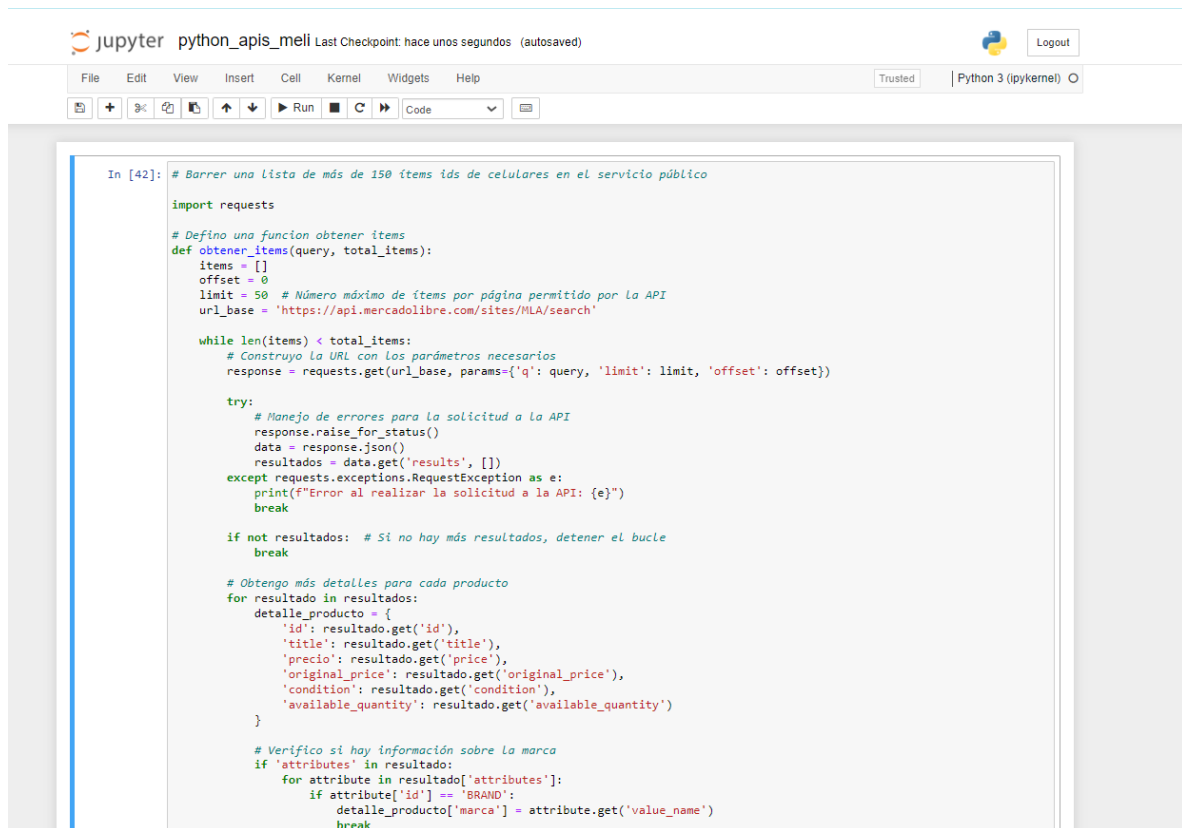
Challenge Engineer - Segunda Parte - Python y APIs

En esta parte del challenge realizare un análisis sobre la oferta/vidriera de las opciones de tres diferentes productos que responden a distintas búsquedas en el sitio Mercadolibre.com.ar utilizando el lenguaje Python y las bibliotecas que considere necesarias.

Punto1: Debo Barrer una lista de más de 150 ítems ids en el servicio público

Esto lo hago con la url: <https://api.mercadolibre.com/sites/MLA/search>

Para esto defino una función llamada obtener_items



```
In [42]: # Barrer una lista de más de 150 ítems ids de celulares en el servicio público

import requests

# Defino una función obtener items
def obtener_items(query, total_items):
    items = []
    offset = 0
    limit = 50 # Número máximo de ítems por página permitido por la API
    url_base = 'https://api.mercadolibre.com/sites/MLA/search'

    while len(items) < total_items:
        # Construyo la URL con los parámetros necesarios
        response = requests.get(url_base, params={'q': query, 'limit': limit, 'offset': offset})

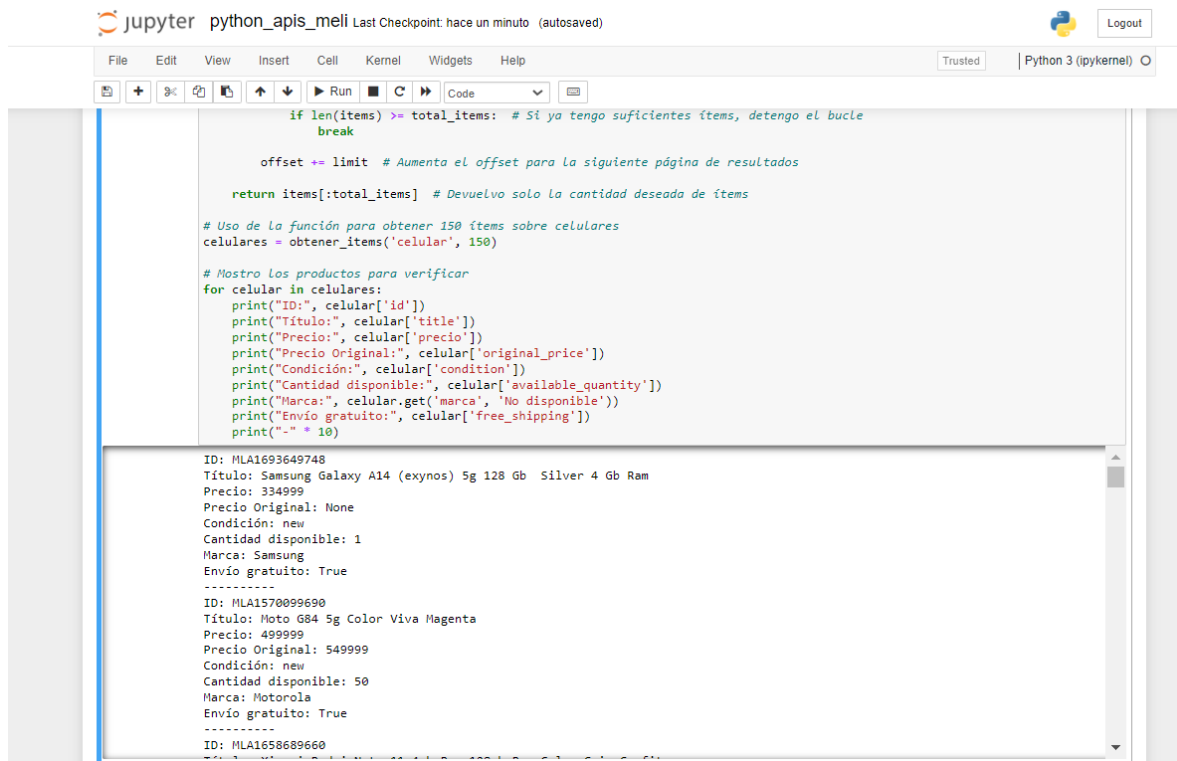
        try:
            # Manejo de errores para la solicitud a la API
            response.raise_for_status()
            data = response.json()
            resultados = data.get('results', [])
        except requests.exceptions.RequestException as e:
            print(f"Error al realizar la solicitud a la API: {e}")
            break

        if not resultados: # Si no hay más resultados, detener el bucle
            break

        # Obtengo más detalles para cada producto
        for resultado in resultados:
            detalle_producto = {
                'id': resultado.get('id'),
                'title': resultado.get('title'),
                'precio': resultado.get('price'),
                'original_price': resultado.get('original_price'),
                'condition': resultado.get('condition'),
                'available_quantity': resultado.get('available_quantity')
            }

            # Verifico si hay información sobre la marca
            if 'attributes' in resultado:
                for attribute in resultado['attributes']:
                    if attribute['id'] == 'BRAND':
                        detalle_producto['marca'] = attribute.get('value_name')
                        break
```

Para este caso muestro la lista de 150 productos de Celulares, con los datos que me interesa obtener como ID, Título, Precio, Precio Anterior, Condición, Cantidad Disponible, Marca y Envío Gratuito.



```
jupyter python_api_meli Last Checkpoint: hace un minuto (autosaved)
File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

if len(items) >= total_items: # Si ya tengo suficientes items, detengo el bucle
    break

    offset += limit # Aumenta el offset para la siguiente página de resultados

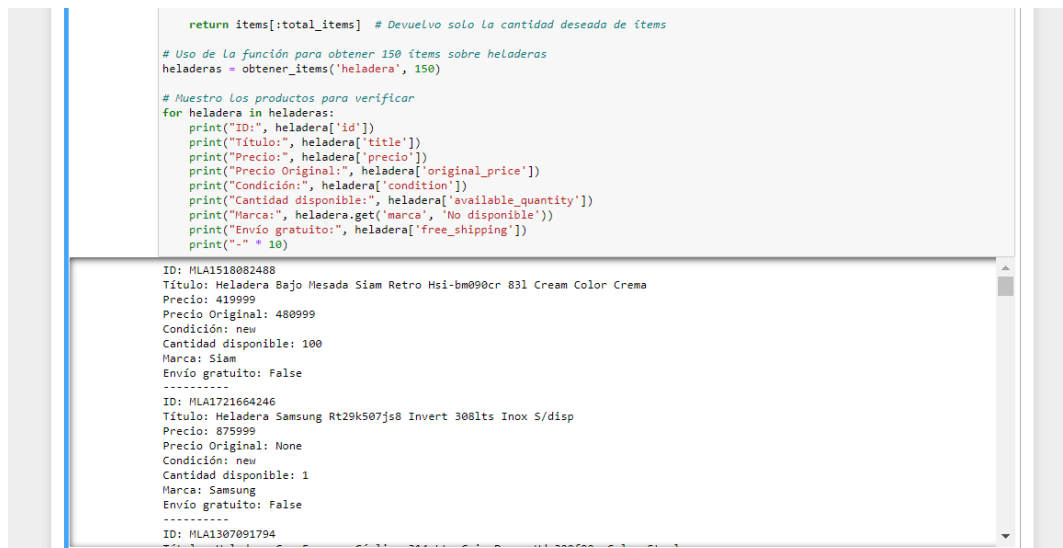
return items[:total_items] # Devuelvo solo la cantidad deseada de items

# Uso de la función para obtener 150 items sobre celulares
celulares = obtener_items('celular', 150)

# Muestro los productos para verificar
for celular in celulares:
    print("ID:", celular['id'])
    print("Título:", celular['title'])
    print("Precio:", celular['precio'])
    print("Precio Original:", celular['original_price'])
    print("Condición:", celular['condition'])
    print("Cantidad disponible:", celular['available_quantity'])
    print("Marca:", celular.get('marca', 'No disponible'))
    print("Envío gratuito:", celular['free_shipping'])
    print("-" * 10)

ID: MLA1693649748
Título: Samsung Galaxy A14 (exynos) 5g 128 Gb Silver 4 Gb Ram
Precio: 334999
Precio Original: None
Condición: new
Cantidad disponible: 1
Marca: Samsung
Envío gratuito: True
-----
ID: MLA1570099690
Título: Moto G84 5g Color Viva Magenta
Precio: 499999
Precio Original: 549999
Condición: new
Cantidad disponible: 50
Marca: Motorola
Envío gratuito: True
-----
ID: MLA1658689660
```

Esto mismo lo hago para 3 productos diferentes, ya que quiero obtener este listado de 150 producto para Celulares, Heladeras y Termos.



```
return items[:total_items] # Devuelvo solo la cantidad deseada de items

# Uso de la función para obtener 150 items sobre heladeras
heladeras = obtener_items('heladera', 150)

# Muestro los productos para verificar
for heladera in heladeras:
    print("ID:", heladera['id'])
    print("Título:", heladera['title'])
    print("Precio:", heladera['precio'])
    print("Precio Original:", heladera['original_price'])
    print("Condición:", heladera['condition'])
    print("Cantidad disponible:", heladera['available_quantity'])
    print("Marca:", heladera.get('marca', 'No disponible'))
    print("Envío gratuito:", heladera['free_shipping'])
    print("-" * 10)

ID: MLA1518082488
Título: Heladera Bajo Mesada Siam Retro Hsi-bm090cr 83l Cream Color Crema
Precio: 419999
Precio Original: 480999
Condición: new
Cantidad disponible: 100
Marca: Siam
Envío gratuito: False
-----
ID: MLA1721664246
Título: Heladera Samsung Rt29k507js8 Invert 308lts Inox 5/disp
Precio: 875999
Precio Original: None
Condición: new
Cantidad disponible: 1
Marca: Samsung
Envío gratuito: False
-----
ID: MLA1307091794
```

```

        if len(items) >= total_items: # Si ya tenemos suficientes ítems, detengo el bucle
            break

        offset += limit # Aumento el offset para la siguiente página de resultados

    return items[:total_items] # Devuelvo solo la cantidad deseada de ítems

# Uso de la función para obtener 150 ítems sobre termnos
termnos = obtener_items('termo', 150)

# Muestro los productos para verificar
for termno in termnos:
    print("ID:", termno['id'])
    print("Título:", termno['title'])
    print("Precio:", termno['precio'])
    print("Precio Original:", termno['original_price'])
    print("Condición:", termno['condition'])
    print("Cantidad disponible:", termno['available_quantity'])
    print("Marca:", termno.get('marca', 'No disponible'))
    print("Envío gratuito:", termno['free_shipping'])
    print("-" * 10)

```

```

ID: MLA1642499266
Título: Termo De Acero Inoxidable 1 Litro Frio Calor
Precio: 23099.34
Precio Original: 34999
Condición: new
Cantidad disponible: 100
Marca: Genérica
Envío gratuito: True
-----
ID: MLA1424303345
Título: Termo Bala Acero Inox Doble Capa Termica 1 Litro Con Estuche Color Plateado
Precio: 9400
Precio Original: None
Condición: new
Cantidad disponible: 1
Marca: Libercam
Envío gratuito: False

```

Punto 2: Ahora para el siguiente caso, por cada resultado, realizar el correspondiente GET por Item_Id al recurso público:

Esto lo hago con la url: https://api.mercadolibre.com/items/{item_id}

Para esto defino una función llamada obtener_detalle_items

```

In [38]: # Realizo el correspondiente GET por Item_Id de celulares al recurso

import requests

# Defino una función obtener detalles items
def obtener_detalle_item(item_id):
    url = f'https://api.mercadolibre.com/items/{item_id}'
    try:
        response = requests.get(url)
        response.raise_for_status() # Manejar errores de solicitud
        data = response.json()
        return data
    except requests.exceptions.RequestException as e:
        print(f"Error al obtener detalles del ítem {item_id}: {e}")
        return None

```

De esta forma obtengo detalles adicionales de los productos en formato JSON, en este caso de celulares

```
# Muestro los productos para verificar
for celular in celulares:
    print("ID:", celular['id'])
    print("Título:", celular['title'])
    print("Precio:", celular['precio'])
    print("Precio Original:", celular['original_price'])
    print("Condición:", celular['condition'])
    print("Cantidad disponible:", celular['available_quantity'])
    print("Marca:", celular.get('marca', 'No disponible'))
    print("Envío gratuito:", celular['free_shipping'])
    print("-" * 10)

# Obtengo detalles adicionales para cada ítem
for celular in celulares:
    item_id = celular['id']
    detalles_item = obtener_detalle_item(item_id)
    if detalles_item:
        print("Detalles adicionales para el ítem", item_id)
        print(detalles_item)
        print("-" * 10)

Envío gratuito: True
-----

Detalles adicionales para el ítem MLA1422976047
{'id': 'MLA1422976047', 'site_id': 'MLA', 'title': 'Celular Quantum Q10 Negro 4gb Ram + 128gb', 'seller_id': 185702920, 'category_id': 'MLA1055', 'official_store_id': None, 'price': 148800, 'base_price': 148800, 'original_price': 193999, 'currency_id': 'ARS', 'initial_quantity': 93, 'sale_terms': [{'id': 'WARRANTY_TYPE', 'name': 'Tipo de garantía', 'value_id': '2230279', 'value_name': 'Garantía de fábrica', 'value_struct': None, 'values': [{'id': '2230279', 'name': 'Garantía de fábrica', 'struct': None}], 'value_type': 'list'}, {'id': 'WARRANTY_TIME', 'name': 'Tiempo de garantía', 'value_id': None, 'value_name': '12 meses', 'value_struct': {'number': 12, 'unit': 'meses'}, 'values': [{'id': None, 'name': '12 meses', 'struct': {'number': 12, 'unit': 'meses'}}, {'value_type': 'number_unit'}], 'buying_mode': 'buy_it_now', 'listing_type_id': 'gold_special', 'condition': 'new', 'permalink': 'https://articulo.mercadolibre.com.ar/MLA-1422976047-celular-quantum-q10-negro-4gb-ram-128gb-_JM', 'thumbnail_id': '777498-MLA75764064328_042024', 'thumbnail': 'http://http2.mlstatic.com/D_777498-MLA75764064328_042024-I.jpg', 'pictures': [{'id': '777498-MLA75764064328_042024', 'url': 'http://http2.mlstatic.com/D_777498-MLA75764064328_042024-I.jpg', 'secure_url': 'https://http2.mlstatic.com/D_777498-MLA75764064328_042024-O.jpg', 'size': '233x500', 'max_size': '490x1051', 'quality': ''}, {'id': '881200-MLA75764409270_042024', 'url': 'http://http2.mlstatic.com/D_881200-MLA75764409270_042024-O.jpg', 'secure_url': 'https://http2.mlstatic.com/D_881200-MLA75764409270_042024-O.jpg', 'size': '231x500', 'max_size': '487x1050', 'quality': ''}], 'video_id': None, 'descriptions': [], 'accepts_mercadopago': True, 'non_mercado_pago_payment_methods': [], 'shipping': {'mode': 'me2', 'methods': [], 'tags': ['self_service_in', 'mandatory_free_shipping'], 'dimensions': None, 'local_pick_up': True, 'free_shipping': True, 'logistic_type': 'cross_docking', 'store_pick_up': False}, 'internal'}
```

También para heladeras

```
# Obtengo detalles adicionales para cada ítem
for heladera in heladeras:
    item_id = heladera['id']
    detalles_item = obtener_detalle_item(item_id)
    if detalles_item:
        print("Detalles adicionales para el ítem", item_id)
        print(detalles_item)
        print("-" * 10)

Condición: new
Cantidad disponible: 1
Marca: Samsung
Envío gratuito: False
-----

Detalles adicionales para el ítem MLA1281067393
{'id': 'MLA1281067393', 'site_id': 'MLA', 'title': 'Heladera Drean Hdr280f00 Blanca Con Freezer 264l 220v Color Blanco', 'seller_id': 192769857, 'category_id': 'MLA398582', 'official_store_id': None, 'price': 486499, 'base_price': 486499, 'original_price': 599999, 'currency_id': 'ARS', 'initial_quantity': 1362, 'sale_terms': [{'id': 'WARRANTY_TYPE', 'name': 'Tipo de garantía', 'value_id': '2230279', 'value_name': 'Garantía de fábrica', 'value_struct': None, 'values': [{'id': '2230279', 'name': 'Garantía de fábrica', 'struct': None}], 'value_type': 'list'}, {'id': 'INVOICE', 'name': 'Facturación', 'value_id': '6891885', 'value_name': 'Factura A', 'value_struct': None, 'values': [{'id': '6891885', 'name': 'Factura A', 'struct': None}], 'value_type': 'list'}, {'id': 'WARRANTY_TIME', 'name': 'Tiempo de garantía', 'value_id': None, 'value_name': '12 meses', 'value_struct': {'number': 12, 'unit': 'meses'}, 'values': [{'id': None, 'name': '12 meses', 'struct': {'number': 12, 'unit': 'meses'}}, {'value_type': 'number_unit'}], 'buying_mode': 'buy_it_now', 'listing_type_id': 'gold_special', 'condition': 'new', 'permalink': 'https://articulo.mercadolibre.com.ar/MLA-1281067393-heladera-drean-hdr280f00-blanca-con-freezer-264l-220v-color-blanco-_JM', 'thumbnail_id': '637136-MLU74118237498_012024', 'thumbnail': 'http://http2.mlstatic.com/D_637136-MLU74118237498_012024-I.jpg', 'pictures': [{'id': '637136-MLU74118237498_012024', 'url': 'http://http2.mlstatic.com/D_637136-MLU74118237498_012024-I.jpg', 'secure_url': 'https://http2.mlstatic.com/D_637136-MLU74118237498_012024-O.jpg', 'size': '176x500'}
```

Y para Termos

```
print('marca:', termo.get('marca', 'no disponible'))
print("Envío gratuito:", termo['free_shipping'])
print("-" * 10)

# Obtener detalles adicionales para cada ítem
for termo in termos:
    item_id = termo['id']
    detalles_item = obtener_detalle_item(item_id)
    if detalles_item:
        print("Detalles adicionales para el ítem", item_id)
        print(detalles_item)
        print("-" * 10)

Marca: Kovea
Envío gratuito: True
-----

Detalles adicionales para el ítem MLA1642499266
{'id': 'MLA1642499266', 'site_id': 'MLA', 'title': 'Termo De Acero Inoxidable 1 Litro Frio Calor', 'seller_id': 1071005257,
 'category_id': 'MLA47769', 'official_store_id': None, 'price': 23099.34, 'base_price': 23099.34, 'original_price': 34999, 'c
urrency_id': 'ARS', 'initial_quantity': 762, 'sale_terms': [{'id': 'INVOICE', 'name': 'Facturación', 'value_id': '6891885',
'value_name': 'Factura A', 'value_struct': None, 'values': [{'id': '6891885', 'name': 'Factura A', 'struct': None}], 'value_
type': 'list'}, {'id': 'WARRANTY_TYPE', 'name': 'Tipo de garantía', 'value_id': '2230280', 'value_name': 'Garantía del vende
dor', 'value_struct': None, 'values': [{'id': '2230280', 'name': 'Garantía del vendedor', 'struct': None}], 'value_type': 'l
ist'}, {'id': 'WARRANTY_TIME', 'name': 'Tiempo de garantía', 'value_id': None, 'value_name': '60 días', 'value_struct': {'nu
mber': 60, 'unit': 'días'}, 'values': [{'id': None, 'name': '60 días', 'struct': {'number': 60, 'unit': 'días'}}, 'value_ty
pe': 'number_unit'}], 'buying_mode': 'buy_it_now', 'listing_type_id': 'gold_special', 'condition': 'new', 'permalink': 'http
s://articulo.mercadolibre.com.ar/MLA-1642499266-termo-de-acero-inoxidable-1-litro-frio-calor-_JM', 'thumbnail_id': '887880-M
LU75074058731_032024', 'thumbnail': 'http://http2.mlstatic.com/D_887880-MLU75074058731_032024-I.jpg', 'pictures': [{'id': '8
87880-MLU75074058731_032024', 'url': 'http://http2.mlstatic.com/D_887880-MLU75074058731_032024-O.jpg', 'secure_url': 'http
s://http2.mlstatic.com/D_887880-MLU75074058731_032024-O.jpg', 'size': '304x500', 'max_size': '730x1199', 'quality': ''}, {'i
d': '886035-MLU71549782778_092023', 'url': 'http://http2.mlstatic.com/D_886035-MLU71549782778_092023-O.jpg', 'secure_url':
'http://http2.mlstatic.com/D_886035-MLU71549782778_092023-O.jpg', 'size': '304x500', 'max_size': '680x680', 'quality': '11
1'}
```

Punto 3: En el siguiente punto tengo que escribir los resultados en un archivo plano delimitado por comas, desnormalizando el JSON obtenido en el paso anterior, en tantos campos como sea necesario para guardar las variables que te interesen modelar.

Esto sería exportar estos datos en un archivo CSV.

Para realizar esto utilizo el objeto `csv.writer` con sus funciones incorporadas.

Hago esto para los tres productos, celulares, heladeras y termos.

De esta forma genero los archivos `celulares.csv`, `heladeras.csv` y `termos.csv`.

```
In [40]: # Exportar los datos de la API de celulares a un archivo CSV para el posterior analisis

import requests
import csv

# Obtengo 150 ítems sobre celulares
celulares = obtener_items('celular', 150)

# Abro archivo CSV para escritura
with open('celulares.csv', 'w', newline='') as csvfile:
    # Creo objeto csv.writer
    writer = csv.writer(csvfile, delimiter=',')

    # Escribo la cabecera del CSV
    columnas = ['ID', 'Título', 'Precio', 'Precio Original', 'Condición', 'Cantidad disponible', 'Marca', 'Envío gratuito']
    writer.writerow(columnas)

    # Escribo los datos de los productos
    for celular in celulares:
        datos_producto = [
            celular['id'],
            celular['title'],
            celular['precio'],
            celular['original_price'],
            celular['condition'],
            celular['available_quantity'],
            celular.get('marca', 'No disponible'),
            celular['free_shipping']
        ]
        writer.writerow(datos_producto)
```

```
In [47]: # Exportar Los datos de La API de heladeras a un archivo CSV para el posterior analisis

import requests
import csv

# Obtener 150 ítems sobre heladeras
heladeras = obtener_items('heladera', 150)

# Abrir archivo CSV para escritura
with open('heladeras.csv', 'w', newline='') as csvfile:
    # Crear objeto csv.writer
    writer = csv.writer(csvfile, delimiter=',')

    # Escribir La cabecera del CSV
    columnas = ['ID', 'Título', 'Precio', 'Precio Original', 'Condición', 'Cantidad disponible', 'Marca', 'Envío gratuito']
    writer.writerow(columnas)

    # Escribir Los datos de Los productos
    for heladera in heladeras:
        datos_producto = [
            heladera['id'],
            heladera['title'],
            heladera['precio'],
            heladera['original_price'],
            heladera['condition'],
            heladera['available_quantity'],
            heladera.get('marca', 'No disponible'),
            heladera['free_shipping']
        ]
        writer.writerow(datos_producto)
```

```
In [48]: # Exportar Los datos de La API de termos a un archivo CSV para el posterior analisis

import requests
import csv

# Obtengo 150 ítems sobre termos
termos = obtener_items('termo', 150)

# Abro archivo CSV para escritura
with open('termos.csv', 'w', newline='') as csvfile:
    # Creo objeto csv.writer
    writer = csv.writer(csvfile, delimiter=',')

    # Escribo La cabecera del CSV
    columnas = ['ID', 'Título', 'Precio', 'Precio Original', 'Condición', 'Cantidad disponible', 'Marca', 'Envío gratuito']
    writer.writerow(columnas)

    # Escribo Los datos de Los productos
    for termo in termos:
        datos_producto = [
            termo['id'],
            termo['title'],
            termo['precio'],
            termo['original_price'],
            termo['condition'],
            termo['available_quantity'],
            termo.get('marca', 'No disponible'),
            termo['free_shipping']
        ]
        writer.writerow(datos_producto)
```

Y compruebo que se me generaron estos archivos.

<input type="checkbox"/> Untitled6.ipynb	Running	hace 9 horas	5.65 kB
<input type="checkbox"/> celulares.csv		hace 19 horas	14 kB
<input type="checkbox"/> Datos - homologados.xlsx		hace un mes	48.7 kB
<input type="checkbox"/> heladeras.csv		hace 19 horas	16.5 kB
<input type="checkbox"/> linear_regression.png		hace 2 meses	34.1 kB
<input type="checkbox"/> termos.csv		hace 19 horas	15.9 kB

Corroboro los archivos creados que tengan los daots de cabecera con los registros, son 150 registros por producto.

[illegible][illegible]

Autoguardado [iconos] términos [icono] Buscar

Archivo Inicio Insertar Disposición de página Fórmulas Datos Revisar Vista Automatizar Ayuda

[iconos] Aptos Narrow 11 [iconos] [iconos] Ajustar texto [iconos] General [iconos] Formato condicional [iconos] Dar formato como tabla [iconos] Estilos de celda

Portapapeles Fuente Alineación Número Estilos

N4 [iconos] [iconos] [iconos]

	A	B	C	D	E	F	G	H	I	J	K	L
1	ID,Título,Precio,Precio Original,Condición,Cantidad disponible,Marca,Envío gratuito											
2	MLA1642499266,Termo De Acero Inoxidable 1 Litro Frio Calor,23099.34,34999,new,100,Genérica,True											
3	MLA1424303345,Termo Bala Acero Inox Doble Capa Termica 1 Litro Con Estuche Color Plateado,9400,,new,1,Libercam,False											
4	MLA1409013755,Termo Thermio 1 L Pico Cebador Uso Extremo De Acero Inoxidable 1l Negro,34999.3,49999,new,100,Thermio,True											
5	MLA1269105647,Termo Media Manija Uruguayo 1 Litro Acero Inoxidable Cosmos Color Plateado,11500,,new,1,Cosmos,False											
6	MLA1489681646,Termo Bala Acero Inoxidable 1 Litro Con Tapón Matero Y Funda Color Plateado,10135,,new,1,Termo,False											
7	MLA1719496810,Termo Waterdog Ombu 1000ml Negro Ombu1000bk,40740,,new,250,Waterdog,True											
8	MLA1379735015,Termo Stanley Clásico 1.4 Litros Con Manija Color Azul Lake,120000,,new,50,Stanley,True											
9	MLA1565666500,Termo Stanley Classic 950ml Rosa,106000,,new,250,Stanley,True											
10	MLA1418126260,Termo South Port Pico Cebador 1 Litro Acero Inoxidable,26499,27999,new,150,South Port,True											
11	MLA1583167802,Termo Kushiros Ds607v De Acero Inoxidable 1l Gris,25999,32999,new,500,Kushiro,True											
12	MLA1696242276,Termo Kanson Acero Ks-009 De Acero Inoxidable 1l Blancoblanco,17100,,new,50,Kanson,False											
13	MLA1670706112,Termo Stanley Original Mate System Classic 1.2 Litros,149999,,new,1,Stanley,True											
14	MLA1758798764,Termo Lumilagro Acero Luminox De Acero Inoxidable 1l Acero,36672,,new,1,Lumilagro,True											
15	MLA1175815562,Termo Acero Inoxidable Lusqtoff 750 MI Frio Calor 20hs Verde,23399.13,29998.89,new,1,Lusqtoff,True											
16	MLA1389251414,Termo Coleman Matero De Acero Inoxidable 0.7l Negro,63543.45,74757,new,1,Coleman,True											
17	MLA1396221554,Termo Bala 1 Litro Acero Inoxidable Doble Capa Termica Color Plateado,10999,12999,new,100,SZM,False											
18	MLA927863836,Stanley Termo Clásico 1.4 Lts Asa Plegable Verde,120000,,new,1,Stanley,True											
19	MLA1394162097,Termo Acero Inoxidable Pico Cebador Dakota 750ml Colores Color Rosa,44981.5,44981.5,new,1,Dakota,True											
20	MLA1419228083,Termo Waterdog Ombu 1000ml Blanco Ombu1000wh,42176,,new,500,Waterdog,True											
21	MLA1408950401,Termo Thermio 1 L Pico Cebador Uso Extremo De Acero Inoxidable 1l Blanco,34999.3,49999,new,50,Thermio,True											
22	MLA1402627507,Termo Acero Inoxidable Pico Cebador Dakota 1 Litro Colores Color Negro,29420.32,51614.6,new,1,Dakota,True											
23	MLA1398464067,Termo Rolan Mate Listo Autocebante 750ml Rosa,15336,,new,1,Rolan,False											
24	MLA1396424161,Termo Media Manija 1 L Acero Inoxidable Pico Matero Cebador Color Plateado,11054,,new,50,Big Star,False											
25	MLA1407777799,Termo Aquatal Everest De Acero Inoxidable 1l Rosa,35991,39990,new,1,Aquatal,True											
26	MLA1390258760,Termo Discovery Verde Acero Inoxidable - Doble Capa Frio Calor Mate Agua 750ml,27649,,new,1,Discovery,True											
27	MLA1693839110,Termo 1 L Media Manija Pico Matero Acero Doble Capa Color Plateado,11881,,new,1,El Rey del Termo,False											

← términos + →

Punto 4: Para el punto 4 elaboro un diseño de la solución, construyo una api en donde divido estas tareas en tres componentes de clases diferentes para consumir la API de Mercado Libre, Exportar los datos a archivo CSV, y se me ocurrió crear una clase más que convierta archivos CSV a archivos Excel, esto lo veo útil por si esta información la tuviera que interactuar con bases de datos en donde tendria las tablas ya construidas. Tambien genero el diagrama de esta solución.

Organizo todo en este directorio.

> Python_y_APIs > diseño_api_punto_cuatro [iconos] Bus

	Nombre	Fecha de modificación	Tipo	Tamaño
	Api_meli	3/5/2024 00:50	Carpeta de archivos	
	Diagrama_Api	3/5/2024 00:44	Archivo PNG	22 KB
	diagrama_meli.drawio	3/5/2024 00:42	Archivo DRAWIO	10 KB

Dentro me encuentro con estas 3 clases.

Python_y_APIs > diseño_api_punto_cuatro > Api_meli >					Buscar en Api_
Nombre		Fecha de modificación	Tipo	Tamaño	
pycache		2/5/2024 22:32	Carpeta de archivos		
DataTransformer		3/5/2024 00:36	Archivo de origen ...	2 KB	
MercadoLibreAPIClient		2/5/2024 22:28	Archivo de origen ...	3 KB	
MercadoLibreDataExtractor		2/5/2024 22:30	Archivo de origen ...	3 KB	

La clase MercadoLibreAPIClient posee tres funciones

```
MercadoLibreAPIClient.py X
MercadoLibreAPIClient.py
1  import requests
2
3  # Defino la clase MercadoLibreAPIClient
4  class MercadoLibreAPIClient:
5
6      # Defino la funcion __init__
7      def __init__(self, base_url):
8          self.base_url = base_url
9
10     # Defino la funcion get_item_details
11     def get_item_details(self, item_id):
12         url = f"{self.base_url}/items/{item_id}"
13         try:
14             response = requests.get(url)
15             response.raise_for_status() # Raise HTTPError para bad responses
16             return response.json()
17         except requests.RequestException as e:
18             print(f"Error al obtener los detalles del artículo: {e}")
19             return None
20
21     # Defino la funcion search_items
22     def search_items(self, query, limit):
23         items = []
24         offset = 0
25         count = 0 # Contador para llevar la cuenta de los artículos recuperados
26         while count < limit:
27             url = f"{self.base_url}/sites/MLA/search?q={query}&limit=50&offset={offset}"
28             try:
29                 response = requests.get(url)
30                 response.raise_for_status() # Raise HTTPError para bad responses
31                 results = response.json()['results']
32                 if not results:
33                     break # No mas resultados
34                 items.extend(results)
35                 offset += 50
36                 count += len(results) # Incremento el contador
37             except requests.RequestException as e:
38                 print(f"Error al buscar artículos: {e}")
39                 return None
40             return items[:limit] # Retorno solo la cantidad especificada de artículos
41
42     # Ejemplo de casos de uso
43     client = MercadoLibreAPIClient("https://api.mercadolibre.com")
44
```

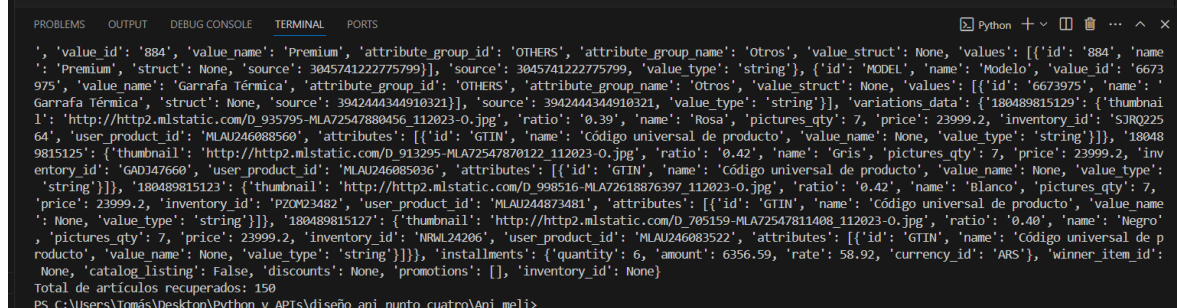
La función `__init__` es un método especial en Python que se llama automáticamente cuando se crea una nueva instancia de una clase. Es decir, cuando se crea un objeto de la clase `MercadoLibreAPIClient`, este método se ejecuta automáticamente para inicializar el objeto.

La función `get_item_details`: Este método toma un `item_id` como entrada y devuelve los detalles del artículo correspondiente, utilizando la URL base proporcionada en la inicialización de la clase. Utiliza la biblioteca `requests` para enviar una solicitud GET a la API de MercadoLibre y luego procesa la respuesta. Si hay algún error durante la solicitud, imprime un mensaje de error y devuelve `None`.

La función `search_items`: Este método toma una consulta (query) y un límite (limit) como entrada y devuelve una lista de artículos que coinciden con la consulta, hasta el límite especificado. Utiliza un bucle `while` para recuperar los artículos en lotes de 50 (que es el máximo permitido por la API de MercadoLibre) hasta que se alcanza el límite especificado. Cada lote de resultados se agrega a una lista llamada `items`. Si no hay más resultados, el bucle se interrumpe. Si ocurre algún error durante la solicitud, imprime un mensaje de error y devuelve `None`.

Después hago los ejemplos de casos de uso de la clase para cada caso particular

```
91 | print("No se pudieron obtener resultados de la búsqueda.")
92
93 | print(f"Total de artículos recuperados: {len(items)}")
```



The screenshot shows a Python IDE with a terminal window. The terminal displays a JSON response from the MercadoLibre API, which includes details about a product (a thermos) and its variations. The response is truncated for brevity. Below the JSON, it shows the total number of items recovered: 150. The terminal also shows the file path: PS C:\Users\Tomás\Desktop\Python y APIs\diseño api punto cuatro\Api meli>

La clase MercadoLibreDataExtractor posee cuatro funciones

```

MercadoLibreAPIClient.py  MercadoLibreDataExtractor.py X
MercadoLibreDataExtractor.py
1  import csv
2  from MercadoLibreAPIClient import MercadoLibreAPIClient
3
4  # Defino la clase MercadoLibreDataExtractor
5  class MercadoLibreDataExtractor:
6
7      # Defino la funcion __init__
8      def __init__(self, api_client):
9          self.api_client = api_client
10
11     # Defino la funcion extract_items_data
12     def extract_items_data(self, query, limit):
13         items = self.api_client.search_items(query, limit)
14         if items:
15             return [self.extract_item_data(item) for item in items]
16         else:
17             return []
18
19     # Defino la funcion extract_item_data
20     def extract_item_data(self, item):
21         return {
22             'ID': item['id'],
23             'Título': item['title'],
24             'Precio': item['price'],
25             'Precio Original': item.get('original_price', 'No disponible'),
26             'Condición': item['condition'],
27             'Cantidad disponible': item['available_quantity'],
28             'Marca': item.get('marca', 'No disponible'),
29             'Envío gratuito': item.get('free_shipping', False) # Agrego un valor predeter
30         }
31
32     def export_to_csv(self, data, filename):
33         try:
34             with open(filename, 'w', newline='', encoding='utf-8-sig') as csvfile:
35                 writer = csv.DictWriter(csvfile, fieldnames=data[0].keys())
36                 writer.writeheader()
37                 writer.writerows(data)
38                 print(f"Los datos se han exportado correctamente a '{filename}'.")
39         except IOError as e:
40             print(f"Error al exportar datos a CSV: {e}")
41
42     # Ejemplo de uso
```

La función `__init__` es el método inicializador de la clase. Recibe un parámetro `api_client`, que es una instancia de la clase `MercadoLibreAPIClient`. Este método asigna el cliente de la API proporcionado a la propiedad `self.api_client`, lo que permite que otros métodos de la clase accedan a los métodos de la API.

La función `extract_items_data` extrae datos de los artículos de MercadoLibre utilizando el cliente de la API. Toma dos parámetros: `query`, que es la consulta de búsqueda, y `limit`, que es el límite de la cantidad de artículos a extraer. Llama al método `search_items` del cliente de la API para obtener una lista de artículos basados en la consulta y el límite. Luego, utiliza el método `extract_item_data` para procesar cada artículo y devuelve una lista de diccionarios con los datos extraídos de los artículos.

La función `extract_item_data` toma un artículo como entrada y extrae datos específicos de ese artículo, como el ID, el título, el precio, etc. Devuelve un diccionario con estos datos.

La función `export_to_csv` exporta los datos extraídos a un archivo CSV. Recibe dos parámetros: `data`, que es una lista de diccionarios con los datos extraídos, y `filename`, que es el nombre del archivo CSV de destino.

Al ejecutar este código en el ejemplo de caso de uso genero los archivos CSV.

```
42 # Ejemplo de uso
43 client = MercadoLibreAPIClient("https://api.mercadolibre.com")
44 extractor = MercadoLibreDataExtractor(client)
45
46 # Extraer datos de los primeros 150 celulares
47 query = "celular"
48 limit = 150
49 items_data = extractor.extract_items_data(query, limit)
50
51 # Exportar datos a un archivo CSV
52 extractor.export_to_csv(items_data, "celularesApi.csv")
53
54 #-----
55
56 # Extraer datos de las primeras 150 heladeras
57 query = "heladera"
58 limit = 150
59 items_data = extractor.extract_items_data(query, limit)
60
61 # Exportar datos a un archivo CSV
62 extractor.export_to_csv(items_data, "heladerasApi.csv")
63
64 #-----
65
66 # Extraer datos de los primeros 150 termos
67 query = "termo"
68 limit = 150
69 items_data = extractor.extract_items_data(query, limit)
70
71 # Exportar datos a un archivo CSV
72 extractor.export_to_csv(items_data, "termosApi.csv")
```

Muestro que ahora se crearon los archivos CSV

The screenshot shows a Python IDE with a file explorer on the left and a code editor on the right. The file explorer shows a project named 'API_MELI' with several CSV files and a Python file named 'MercadoLibreDataExtractor.py'. The code editor displays the implementation of the 'MercadoLibreDataExtractor' class, which uses the 'MercadoLibreAPIClient' to search for items and extract their details. The output pane at the bottom shows the results of the extraction, including item details like ID, title, price, condition, and shipping information, as well as a summary of the total items recovered and the files where the data was exported.

```

1  import csv
2  from MercadoLibreAPIClient import MercadoLibreAPIClient
3
4  # Defino la clase MercadoLibreDataExtractor
5  class MercadoLibreDataExtractor:
6
7      # Defino la funcion __init__
8      def __init__(self, api_client):
9          self.api_client = api_client
10
11     # Defino la funcion extract_items_data
12     def extract_items_data(self, query, limit):
13         items = self.api_client.search_items(query, limit)
14         if items:
15             return [self.extract_item_data(item) for item in items]
16         else:
17             return []
18
19     # Defino la funcion extract_item_data
20     def extract_item_data(self, item):
21         return {
22             'ID': item['id'],
23             'Título': item['title'],
24             'Precio': item['price'],
25             'Precio Original': item.get('original_price', 'No disponible'),
26             'Condición': item['condition'],
27             'Cantidad disponible': item['available_quantity'],
28             'Marca': item.get('marca', 'No disponible'),
29             'Envío gratuito': item.get('free_shipping', False) # Agrego un valor predeterminado

```

pe': 'string'}, {'id': 'ITEM_CONDITION', 'name': 'Condición del item', 'value_id': '2230284', 'value_name': 'Otros', 'value_struct': {'number': 34.8, 'unit': 'cm', 'values': [{'id': 'None', 'name': '34.8', 'value_type': 'number_unit'}, {'id': 'PACKAGE_WEIGHT', 'name': 'Peso del paquete', 'value_id': '2421084', 'value_name': 'Otros', 'value_struct': {'number': 820, 'unit': 'g', 'values': [{'id': 'None', 'name': '820 g', 'value_type': 'number_unit'}, {'id': 'THERMO_WEIGHT', 'name': 'Peso del termo', 'value_id': '2421084', 'value_name': 'Otros', 'value_struct': {'number': 939, 'unit': 'g', 'values': [{'id': '24210844', 'name': '939 g', 'value_type': 'number_unit'}, {'id': 'INSTALLMENTS', 'name': 'Plazos de pago', 'value_id': '24210844', 'value_name': 'Otros', 'value_struct': {'number': 6, 'unit': 'months', 'values': [{'id': 'None', 'name': '6 meses', 'value_type': 'number_unit'}, {'id': 'WINNER_ITEM_CURRENCY', 'name': 'Moneda ganadora', 'value_id': '24210844', 'value_name': 'Otros', 'value_struct': {'number': 4046, 'unit': 'ARS', 'values': [{'id': 'None', 'name': '4046 ARS', 'value_type': 'number_unit'}]}}]}}]}}]}}]

Total de artículos recuperados: 150
 Los datos se han exportado correctamente a 'celularesApi.csv'.
 Los datos se han exportado correctamente a 'heladerasApi.csv'.
 Los datos se han exportado correctamente a 'termosApi.csv'.
 PS C:\Users\Tomás\Desktop\Python y APIs\diseño api punto cuatro\api meli>

Finalmente tengo la clase DataTransformer que posee el metodo transform_to_excel

```

MercadoLibreAPIClient.py  MercadoLibreDataExtractor.py  DataTransformer.py X
DataTransformer.py
1  import csv
2  import json
3  import pandas as pd
4
5  # Defino la clase DataTransformer
6  class DataTransformer:
7
8      # Defino la funcion transform_to_excel
9      def transform_to_excel(self, csv_file, excel_file):
10         try:
11             df = pd.read_csv(csv_file)
12             df.to_excel(excel_file, index=False)
13             print(f"Los datos se han transformado y guardado en {excel_file}.")
14         except Exception as e:
15             print(f"Error al transformar datos a formato Excel: {e}")
16
17     # Ejemplo de uso
18     if __name__ == "__main__":
19
20         # Nombre del archivo CSV de entrada
21         csv_file1 = "celularesApi.csv"
22
23         # Nombre del archivo CSV de entrada
24         csv_file2 = "heladerasApi.csv"
25
26         # Nombre del archivo CSV de entrada
27         csv_file3 = "terminosApi.csv"
28
29         # Nombre del archivo Excel de salida
30         excel_file1 = "productosTransform1.xlsx"
31
32         # Nombre del archivo Excel de salida
33         excel_file2 = "productosTransform2.xlsx"
34
35         # Nombre del archivo Excel de salida
36         excel_file3 = "productosTransform3.xlsx"
37
38         # Crear instancia del transformador de datos
39         transformer = DataTransformer()
40
41         # Transformar y guardar los datos en un archivo Excel
42         transformer.transform_to_excel(csv_file1, excel_file1)
43
44         # Transformar y guardar los datos en un archivo Excel
45         transformer.transform_to_excel(csv_file2, excel_file2)
46

```

Esta clase tiene el metodo transform_to_excel que toma la ruta de un archivo CSV como entrada y transforma los datos en ese archivo en un formato Excel utilizando pandas. Luego, guarda el resultado en un archivo Excel especificado por la ruta proporcionada. Si ocurre algún error durante el proceso de transformación o guardado, se captura y se imprime un mensaje de error.

Al ejecutar veo que se me generan los archivos nuevos en Excel.

```

DataTransformer.py
1 import csv
2 import json
3 import pandas as pd
4
5 # Defino la clase DataTransformer
6 class DataTransformer:
7
8     # Defino la funcion transform_to_excel
9     def transform_to_excel(self, csv_file, excel_file):
10
11         try:
12             df = pd.read_csv(csv_file)
13             df.to_excel(excel_file, index=False)
14             print(f"Los datos se han transformado y guardado en {excel_file}.")
15         except Exception as e:
16             print(f"Error al transformar datos a formato Excel: {e}")
17
18 # Ejemplo de uso
19 if __name__ == "__main__":
20
21     # Nombre del archivo CSV de entrada
22     csv_file1 = "celularesApi.csv"
23
24     # Nombre del archivo CSV de entrada
25     csv_file2 = "heladerasApi.csv"
26
27     # Nombre del archivo CSV de entrada
28     csv_file3 = "termosApi.csv"
29
30     # Nombre del archivo Excel de salida

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```

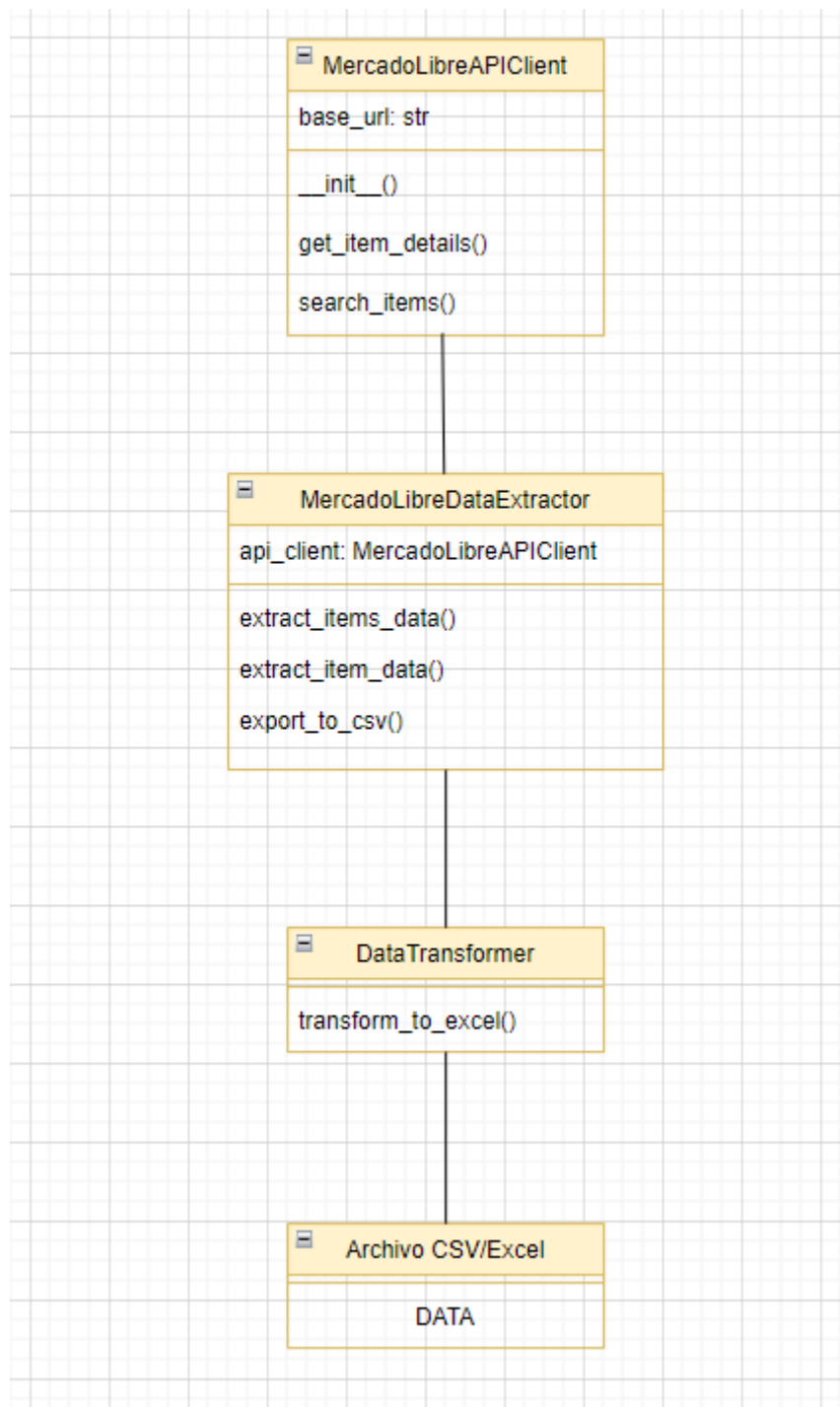
roup_name': 'Otros', 'value_struct': {'number': 34.8, 'unit': 'cm'}, 'values': [{'id': None, 'name': '4333789534002961', 'value_type': 'number_unit'}, {'id': 'PACKAGE_WEIGHT', 'name': 'Peso del paquete', 'value_type': 'number_unit'}, {'id': 'THERMO_WEIGHT', 'name': 'Peso del termo', 'value_id': '61', 'value_struct': {'number': 820, 'unit': 'g'}, 'values': [{'id': None, 'name': '820 g', 'value_type': 'number_unit'}, {'id': 'THERMO_WEIGHT', 'name': 'Peso del termo', 'value_id': '61', 'value_struct': {'number': 939, 'unit': 'g'}, 'values': [{'id': '24210844', 'name': '939 g', 'installments': {'quantity': 6, 'amount': 20262.07, 'rate': 58.92, 'currency_id': 'ARS'}, 'winner': 'CYY44046'}]}]}
Total de artículos recuperados: 150
Los datos se han exportado correctamente a 'celularesApi.csv'.
Los datos se han exportado correctamente a 'heladerasApi.csv'.
Los datos se han exportado correctamente a 'termosApi.csv'.
PS C:\Users\Tomás\Desktop\Python_y_APIs\diseño_api_punto_cuatro\Api_meli> & D:/Anacondist/python...
Los datos se han transformado y guardado en productosTransform1.xlsx.
Los datos se han transformado y guardado en productosTransform2.xlsx.
Los datos se han transformado y guardado en productosTransform3.xlsx.
PS C:\Users\Tomás\Desktop\Python_y_APIs\diseño_api_punto_cuatro\Api_meli>

```

Ejemplo:

	A	B	C	D	E	F	G	H	I
	ID	Título	Precio	Precio Origin	Condición	Disponibilidad	Marca	Envío gratuito	
1	MLA14182	Tcl 40 Nxt	326399	419999	new	1	No disponible	FALSO	
2	MLA15700	Moto G84	499999	549999	new	1	No disponible	FALSO	
3	MLA13993	Tcl 40 Se 1	209999	243799	new	1	No disponible	FALSO	
4	MLA14712	Motorola	229999	249999	new	500	No disponible	FALSO	
5	MLA16994	Kodak Smart	54500		new	50	No disponible	FALSO	
6	MLA17533	Samsung G	319999	579999	new	50	No disponible	FALSO	
7	MLA16994	Smartphon	65990		new	100	No disponible	FALSO	
8	MLA13304	Samsung G	284999		new	250	No disponible	FALSO	
9	MLA16586	Xiaomi Red	202000	249999	new	50	No disponible	FALSO	
10	MLA13175	Samsung G	314999		new	250	No disponible	FALSO	
11	MLA16480	Motorola	600000		new	200	No disponible	FALSO	

Confecciono el Diagrama de esta solución, en Draw.io



Enlace:

https://viewer.diagrams.net/?tags=%7B%7D&highlight=0000ff&edit=_blank&layers=1&nav=1&title=diagrama_meli.drawio#R7Vpbc9o6EP41zJzzkl5tYQceuST0kpwhTU7SPjHCFqBGWFQWAFlrK9kytpHB5pa0UnMhF2tJGu%2FT7tayxXQGs87DE5Gt9RDpGIZ3rwC2hXLMoEJxD%2BpWUSaS8OJFEOGPWWUKO7xK1JKQ2mn2ENBxpBTSjieZJUu9X3k8owOMkZnWbMBJdLZJ3CINMW9C4mufcleH0XamnWZ6D8iPBzFM5tOPWoZw9hYrSQYQY%2FOUipwVQEtRimPfo3nLUSk82K%2FPH1aPJGbZ6fz%2BS74Cf9vfn47%2FEiGux6my7LJTDk852Hfn0eXH98rP74Nule382MTqP%2F%2BUJ1MV4gmSp%2F3SLminXe4D5Dje6nFsFy0sgBfBF7NZjhMYG%2BkJoD6vN71WIIgRI89MVvV%2FRDTCheEONYANJQDZxOhNYdYeLdWAWdyiUFHLrPsdQcUYZfxbCQiCZTKEQz44pblpOxuJc91dQMBcKmG%2FvJXKpuYMCVjUsJgZMA98MHliZjylbYb1LO6TgeiE59D3lKWglfCpzR5yWVZP8BJqRFCWWha8BgMLBcd2mZavGcvmPLpy%2BJp8Jd%2Bg%2FNU2xW%2BHYPQHSPOFsJEtVp1hafaq1UlzhLim47SjdKkt5USqs02XA69nO2r2JzQHwqvJdOB7HRm2fksJzsdJllpPuSoKf0epGksfqRWmqhCcm9BdFMjeh8GqDdlYpRGyG2mkVx4naciTdCAr6VzMIEu9oc3oU27mmi%2BqmVLFrv9BySk0gh7HvJDqnHIYX%2B5fSYU%2Bzz0i90Uf8J7LeODXbHFA7WEbCay%2BJPmjLeoLx4f4pBMSFB9hiTdc2i2MQYU02yRhW9bmNOsyuC7LZhVDUwNO4LD4DTi4ySGrG7cAmDHail5XlzkGwS6fWFqaAMdbZCDLIF9RL0wBxTOT6LbFcQLwL1ENHmMDSwrXI0qO3PgjvmX3eAPzUn4HYyaNZnnR%2BNnNzV62Ef817vn39PdC9r4JTFdS2ITsmQDTZkiL1Q1APzEPEe5mjc8xCHmARnNEujWX9vNIGGZoAgc0choGckyYnpWm8IZf6J6XJjbdAWHr2aC5e4nOpHp3N98FvWB072wJ4X%2FZfl7gEKBNv%2BgwqEmkZ3OME9N6p%2FRY3QsioNo1x1fBpBrTCCLSPepVdLt%2BOVjlYeol6lw57lw7bEyGndsglwGfQh3we6CUkipJZdFLpecL%2FJ3tcOQKgZTPJ8Xa2vRHRM6BbAlp7w1CdXxjqoRrNpSt6nPbc4OVkwDylmIjwXV9NmCXj7dHQ1OOtLCAeGPSDAWVjUQjIqYFpLN1wriYOWk0UsGp9gVH%2BtqHs2wcHnFAxkR%2B79HeT52PmngxdH1b2vqg41htqU3%2B5yeOIJ7MYmrul%2FK15bAc8QXW3HX84QPU01mAit7xQoWzdP4pBrySm5wz2R2Ywp2xK%2BQsymF5OnTPYMTKY%2BZunMEc%2FuDceGkUpqyCYnVbSKoDw%2FbOWfpOj4Yd8ryG%2FthOST1e3dT8C2ewxTcPCM8q8buyDIX2PG3ZXsTSHPNUNyF9T7UknaQQ94meE3naN37lrkYCOmUuKjpvWIZil0O0S32dwtTOgTTWMUQgxy%2FZRwxIHl1J6yRxVGsrlVacOOIhonWqXglbtIHAYgVQdfVeJ3KENTDBsop%2ByXJmYvZomabiNkft2Gibaw5MW3LxNWBbOeNmVg%2FFhOrtXqai8aHugOK%2BCilLmJYrEpmzUMTbuMr4ULCgXcl3GWWJwBc7ki4lYGsg4U%2BISYfgUfmyaf04OoX

Punto 5: Análisis exploratorio

Finalmente realizo un analisis exploratorios con estos datasets con los que vengo generando y trabajando.

Confeccioné tres analisis exploratorios, uno para los productos de celulares, otro para los productos de heladeras y otro para los productos de termos.

La explicación y la documentación detallada está muy bien escrita dentro de estos archivos, por lo que ahí esta toda la información requerida.

Punto 6 (valor agregado por mi): Conclusiones generales de estos Challenge

Quiero dejar unos comentarios acerca de este Challenge de Meli que realicé, la verdad que disfruté mucho hacerlo. A mi me gusta lo que es el analisis de datos, el enfrentar el desafio de resolver las peticiones y ser creativo a la hora de resolverlo.

Aproveché el feriado del dia del trabajador para dedicarle todo el día a avanzar con los puntos, y lograr entregar los resultados lo antes posible.

Encaré este challenge como un proyecto que me dejó muy buenas sensaciones, realmente me gustó mucho los resultados que obtuve de cada instancia, y espero de corazón que les guste tambien el trabajo realizado.

Esfuerzo, dedicación y pasión son las bases con las que encaré este challenge y espero que se haya podido ver reflejado en cada una de las instancias.

Gracias y saludos cordiales !

Tomás Sezaro.