# powerbi\_api.py

import requests

import json

import msal

import pandas as pd

import os

from datetime import datetime

class PowerBIAPI:

def \_\_init\_\_(self, client\_id, client\_secret, tenant\_id):

"""

Inicializa la conexión a la API de Power BI

Args:

client\_id (str): ID de cliente de la aplicación registrada en Azure

client\_secret (str): Secreto de cliente de la aplicación

tenant\_id (str): ID del inquilino de Azure AD

"""

self.client\_id = client\_id

self.client\_secret = client\_secret

self.tenant\_id = tenant\_id

self.base\_url = "https://api.powerbi.com/v1.0/myorg"

self.access\_token = None

def get\_access\_token(self):

"""Obtiene un token de acceso para la API de Power BI"""

authority = f"https://login.microsoftonline.com/{self.tenant\_id}"

app = msal.ConfidentialClientApplication(

client\_id=self.client\_id,

client\_credential=self.client\_secret,

authority=authority

)

# Define los ámbitos necesarios para Power BI

scopes = ["https://analysis.windows.net/powerbi/api/.default"]

# Adquiere token para la aplicación cliente

result = app.acquire\_token\_for\_client(scopes=scopes)

if "access\_token" in result:

self.access\_token = result["access\_token"]

return self.access\_token

else:

error\_message = f"Error al obtener token: {result.get('error')}"

error\_description = result.get('error\_description', '')

raise Exception(f"{error\_message}: {error\_description}")

def \_get\_headers(self):

"""Genera los encabezados HTTP para las solicitudes"""

if not self.access\_token:

self.get\_access\_token()

return {

"Authorization": f"Bearer {self.access\_token}",

"Content-Type": "application/json"

}

def list\_datasets(self):

"""Lista todos los conjuntos de datos disponibles"""

endpoint = f"{self.base\_url}/datasets"

response = requests.get(endpoint, headers=self.\_get\_headers())

if response.status\_code == 200:

return response.json()["value"]

else:

raise Exception(f"Error {response.status\_code}: {response.text}")

def get\_dataset\_by\_name(self, dataset\_name):

"""Busca un conjunto de datos por su nombre"""

datasets = self.list\_datasets()

for dataset in datasets:

if dataset["name"] == dataset\_name:

return dataset

return None

def list\_reports(self):

"""Lista todos los informes disponibles"""

endpoint = f"{self.base\_url}/reports"

response = requests.get(endpoint, headers=self.\_get\_headers())

if response.status\_code == 200:

return response.json()["value"]

else:

raise Exception(f"Error {response.status\_code}: {response.text}")

def get\_report\_data(self, report\_id):

"""Obtiene los datos de un informe específico"""

endpoint = f"{self.base\_url}/reports/{report\_id}"

response = requests.get(endpoint, headers=self.\_get\_headers())

if response.status\_code == 200:

return response.json()

else:

raise Exception(f"Error {response.status\_code}: {response.text}")

def get\_dataset\_tables(self, dataset\_id):

"""Lista todas las tablas en un conjunto de datos"""

endpoint = f"{self.base\_url}/datasets/{dataset\_id}/tables"

response = requests.get(endpoint, headers=self.\_get\_headers())

if response.status\_code == 200:

return response.json()["value"]

else:

raise Exception(f"Error {response.status\_code}: {response.text}")

def query\_dataset(self, dataset\_id, query):

"""Ejecuta una consulta DAX en un conjunto de datos"""

endpoint = f"{self.base\_url}/datasets/{dataset\_id}/executeQueries"

payload = {

"queries": [

{

"query": query

}

],

"serializerSettings": {

"includeNulls": True

}

}

response = requests.post(

endpoint,

headers=self.\_get\_headers(),

data=json.dumps(payload)

)

if response.status\_code == 200:

return response.json()["results"][0]["tables"][0]

else:

raise Exception(f"Error {response.status\_code}: {response.text}")

def export\_report\_to\_file(self, report\_id, file\_format="PDF"):

"""Exporta un informe a un archivo (PDF, PPTX, etc.)"""

supported\_formats = ["PDF", "PPTX", "PNG", "XLSX"]

if file\_format not in supported\_formats:

raise ValueError(f"Formato no soportado. Use uno de: {supported\_formats}")

endpoint = f"{self.base\_url}/reports/{report\_id}/ExportTo"

payload = {

"format": file\_format

}

# Iniciar la exportación

response = requests.post(

endpoint,

headers=self.\_get\_headers(),

data=json.dumps(payload)

)

if response.status\_code == 202: # Accepted

# Obtenemos la URL del estado de la exportación

export\_url = response.headers["Location"]

# Verificamos el estado periódicamente

max\_attempts = 10

for attempt in range(max\_attempts):

status\_response = requests.get(

export\_url,

headers=self.\_get\_headers()

)

if status\_response.status\_code == 200:

status = status\_response.json()

if status["status"] == "Succeeded":

# Descargar el archivo

file\_url = status["resourceLocation"]

file\_response = requests.get(

file\_url,

headers=self.\_get\_headers()

)

# Guardar el archivo localmente

timestamp = datetime.now().strftime("%Y%m%d\_%H%M%S")

filename = f"report\_{report\_id}\_{timestamp}.{file\_format.lower()}"

with open(filename, "wb") as f:

f.write(file\_response.content)

return f"Informe exportado como: {filename}"

elif status["status"] == "Failed":

raise Exception(f"Error al exportar informe: {status.get('error', 'Unknown error')}")

# Esperar antes de verificar de nuevo

import time

time.sleep(2)

raise Exception("Tiempo de espera agotado para la exportación")

else:

raise Exception(f"Error {response.status\_code}: {response.text}")

def query\_to\_dataframe(self, dataset\_id, query):

"""Ejecuta una consulta DAX y devuelve un DataFrame de pandas"""

result = self.query\_dataset(dataset\_id, query)

# Extraer datos y convertir a dataframe

columns = [col['name'] for col in result['columns']]

rows = [list(row.values()) for row in result['rows']]

return pd.DataFrame(rows, columns=columns)

# Ejemplo de uso

if \_\_name\_\_ == "\_\_main\_\_":

# Estas credenciales deberían almacenarse de forma segura

# Lo ideal es usar variables de entorno o un archivo de configuración seguro

client\_id = os.environ.get("POWERBI\_CLIENT\_ID", "TU\_CLIENT\_ID")

client\_secret = os.environ.get("POWERBI\_CLIENT\_SECRET", "TU\_CLIENT\_SECRET")

tenant\_id = os.environ.get("POWERBI\_TENANT\_ID", "TU\_TENANT\_ID")

try:

# Inicializar la conexión a Power BI

pb\_api = PowerBIAPI(client\_id, client\_secret, tenant\_id)

# Listar datasets disponibles

print("Datasets disponibles:")

datasets = pb\_api.list\_datasets()

for dataset in datasets:

print(f"- {dataset['name']} (ID: {dataset['id']})")

if datasets:

# Seleccionar el primer dataset para el ejemplo

dataset\_id = datasets[0]['id']

# Listar tablas del dataset

print("\nTablas en el dataset:")

tables = pb\_api.get\_dataset\_tables(dataset\_id)

for table in tables:

print(f"- {table['name']}")

# Ejecutar una consulta DAX simple

if tables:

table\_name = tables[0]['name']

print(f"\nEjecutando consulta en la tabla {table\_name}:")

query = f"EVALUATE TOPN(10, '{table\_name}')"

try:

# Ejecutar consulta y mostrar resultados como DataFrame

df = pb\_api.query\_to\_dataframe(dataset\_id, query)

print("\nResultados de la consulta:")

print(df.head())

# Guardar resultados a CSV

csv\_file = f"