from fastapi import FastAPI, File, UploadFile, Form

import zipfile

import os

import pandas as pd

from io import BytesIO

import xml.etree.ElementTree as ET

app = FastAPI()

def formatar\_issn(issn: str) -> str:

"""Formata o ISSN no formato XXXX-XXXX."""

issn = issn.replace("-", "").strip()

if len(issn) == 8:

return f"{issn[:4]}-{issn[4:]}"

return issn

def normalizar\_issn(issn: str) -> list:

"""Converte múltiplos ISSNs em uma lista normalizada."""

return [i.strip().replace("-", "") for i in issn.split(',')]

def formatar\_doi(doi: str) -> str:

"""Formata o DOI no formato http://dx.doi.org/xxxxx."""

if doi.lower().startswith("http://") or doi.lower().startswith("https://"):

return doi.strip()

return f"http://dx.doi.org/{doi.strip()}"

def formatar\_jif(jif: float) -> str:

"""Converte o ponto decimal para vírgula."""

return f"{jif:.2f}".replace('.', ',')

def processar\_zip(contents, temp\_folder):

"""Descompacta o ZIP e localiza o arquivo XML."""

try:

with zipfile.ZipFile(BytesIO(contents), 'r') as zip\_ref:

zip\_ref.extractall(temp\_folder)

xml\_files = [f for f in os.listdir(temp\_folder) if f.endswith('.xml')]

if not xml\_files:

return None, "Nenhum arquivo XML encontrado no ZIP."

xml\_path = os.path.join(temp\_folder, xml\_files[0])

return xml\_path, "Arquivo XML extraído com sucesso."

except Exception as e:

return None, f"Erro ao descompactar o arquivo ZIP: {e}"

def processar\_xml(xml\_path, ano\_inicio, ano\_fim):

"""Extrai os dados dos artigos do XML."""

try:

tree = ET.parse(xml\_path)

root = tree.getroot()

artigos = []

for artigo in root.findall('.//ARTIGO-PUBLICADO'):

titulo = artigo.find('DADOS-BASICOS-DO-ARTIGO').get('TITULO-DO-ARTIGO', 'Sem Título')

ano = int(artigo.find('DADOS-BASICOS-DO-ARTIGO').get('ANO-DO-ARTIGO', '0'))

if ano\_inicio <= ano <= ano\_fim:

autores = '; '.join([autor.get('NOME-COMPLETO-DO-AUTOR') for autor in artigo.findall('AUTORES')])

issn = formatar\_issn(artigo.find('DETALHAMENTO-DO-ARTIGO').get('ISSN', 'Sem ISSN'))

doi = formatar\_doi(artigo.find('DADOS-BASICOS-DO-ARTIGO').get('DOI', 'Sem DOI'))

artigos.append({

'Título': titulo,

'Autores': autores,

'Ano': ano,

'ISSN': issn,

'DOI': doi

})

return pd.DataFrame(artigos).sort\_values(by='Ano')

except Exception as e:

print(f"Erro ao processar o XML: {e}")

return pd.DataFrame()

def buscar\_qualis(issn, qualis\_df):

"""Busca a classificação Qualis pelo ISSN."""

issn\_cleaned = issn.replace("-", "").strip()

qualis\_df['ISSN'] = qualis\_df['ISSN'].str.replace("-", "").str.strip()

qualis\_row = qualis\_df[qualis\_df['ISSN'] == issn\_cleaned]

return qualis\_row.iloc[0]['Estrato'] if not qualis\_row.empty else "Não encontrado"

def buscar\_jcr(issn, jcr\_df):

"""Busca o JCR e Quartil pelo ISSN ou eISSN."""

issn\_cleaned = issn.replace("-", "").strip()

jcr\_df['ISSN'] = jcr\_df['ISSN'].astype(str).str.replace("-", "").str.strip()

jcr\_df['eISSN'] = jcr\_df['eISSN'].astype(str).str.replace("-", "").str.strip()

jcr\_row = jcr\_df[(jcr\_df['ISSN'] == issn\_cleaned) | (jcr\_df['eISSN'] == issn\_cleaned)]

if not jcr\_row.empty:

return formatar\_jif(jcr\_row.iloc[0]['2023 JIF']), jcr\_row.iloc[0]['JIF Quartile']

return "Não encontrado", "Não encontrado"

def buscar\_sjr(issn, sjr\_df):

"""Busca o SJR e Quartil pelo ISSN."""

issn\_cleaned = issn.replace("-", "").strip()

sjr\_df['ISSN\_list'] = sjr\_df['Issn'].apply(normalizar\_issn)

sjr\_row = sjr\_df[sjr\_df['ISSN\_list'].apply(lambda x: issn\_cleaned in x)]

if not sjr\_row.empty:

return sjr\_row.iloc[0]['SJR'], sjr\_row.iloc[0]['SJR Best Quartile']

return "Não encontrado", "Não encontrado"

@app.post("/upload/")

async def upload\_file(

file: UploadFile = File(...),

qualis\_file: UploadFile = File(...),

jcr\_file: UploadFile = File(...),

sjr\_file: UploadFile = File(...),

ano\_inicio: int = Form(2000),

ano\_fim: int = Form(2024)

):

"""Processa os arquivos e retorna os artigos com Qualis, JCR e SJR."""

contents = await file.read()

temp\_folder = "./temp"

os.makedirs(temp\_folder, exist\_ok=True)

xml\_path, msg = processar\_zip(contents, temp\_folder)

if not xml\_path:

return {"message": msg}

artigos\_df = processar\_xml(xml\_path, ano\_inicio, ano\_fim)

if artigos\_df.empty:

return {"message": "Nenhum artigo encontrado no período especificado."}

qualis\_df = pd.read\_csv(BytesIO(await qualis\_file.read()), delimiter=',')

jcr\_df = pd.read\_excel(BytesIO(await jcr\_file.read()), engine='openpyxl')

sjr\_df = pd.read\_csv(BytesIO(await sjr\_file.read()), delimiter=';', on\_bad\_lines='skip', engine='python')

artigos\_df['Qualis'] = artigos\_df['ISSN'].apply(lambda x: buscar\_qualis(x, qualis\_df))

artigos\_df[['2023 JIF', 'JIF Quartile']] = artigos\_df['ISSN'].apply(lambda x: pd.Series(buscar\_jcr(x, jcr\_df)))

artigos\_df[['SJR', 'SJR Best Quartile']] = artigos\_df['ISSN'].apply(lambda x: pd.Series(buscar\_sjr(x, sjr\_df)))

return artigos\_df.to\_dict(orient="records")