Outra coisa que notei, a pontuação final é dividida pelo numero de docentes permanentes (Num\_DP), não consigo conferir se a pontuação final

Certo, vamos lá. Ao abrir o ppgec.db:

- Preciso que ele extrai o DOI

- Preciso do ano do artigo

http://dx.doi.org/10.1007/s41024-024-00417-3

Ok, mas lembre que

- a interface 1 não está do jeito que eu quero

- Quero uma coluna na interface 2 depois da coluna 6 (Qualis) que seria editável por enquanto para eu colocar o valor do JCR

- Que o caminho de subpastas e arquivo seja o que eu já falei inúmeras vezes

- Analisando o codigo original, veja qual formatação do ISSN eu quero, que quero virgula e não ponto como separador decimal

- Que na coluna DOI, você já conseguiu que inves de colocar o DOI, colocar um uma seta azul com o link do DOI utilizando o prefixo antes do formato https://doi.org/. Assim, ao clicar na seta azul iremos direto para o artigo.

- E por fim sempre lembre que ainda vamos para interface 3 e uma vez tudo ajustado vou querer um php com designe moderno e intuitivo.

Melhorou, vamos fazer ajustes finos que parecem ser no arquivo processar\_zip.py. Seguem:

- Na coluna DOI, você já conseguiu que inves de colocar o DOI, colocar um uma seta azul com o link do DOI utilizando o prefixo antes do formato https://doi.org/. Assim, ao clicar na seta azul iremos direto para o artigo.

- A coluna Ponderação qualis está errada, você sabe o peso de cada Qualis, coloque o certo.

- Nessa Interface 2, tem que ter um campo de entrada acima da tabela com a ponderação discente, ao digitar nesse campo, a ultima coluna "Ponderação Discente" assume esse valor. Mas, pode deixar como valor de entrada padrão nesse campo editável 0,75.

- Ao lado da coluna "Ponderação Qualis" teria a coluna JCR que veremos no futuro como resolver.

Agora é só multiplicar o valor obtido de acordo com a seguinte regra:

- Quando Num\_DISC = 0, multiplicar pela "Ponderação Discente" definida no campo editável.

Quando Num\_DISC ≠ 0, multiplicar por 1.

Vamos ajustar e para facilitar, vamos dividir a interface em 3 passos:  
  
- Passo 1 – Interface 1 - o ajuste vai ser retirar as ponderações do Qualis e do discente desta interface. Assumindo esses valores ponderação qualis com fixos. O valor da ponderação discente continua sendo edital e ficará no passo 2 – interface 2.

- Passo 2 – interface – 2 – Ajuste da tabela, o ordem das colunas devem ser DOI – Ano – Título - Autores - ISSN – Qualis - Num\_DP - Num\_DISC - Ponderação Qualis - Ponderação Discente. Ao apertar o botão “salvar alterações” iremos para o passo 3 – interface 3.

- Passo 3 – interface – 3 – Tabela final – de acordo com os dados salvos, construa a seguinte tabela: comece criando 19 colunas: "Docente", "Ano", "A1" a "C" (representando categorias de publicações), "Total", e as colunas de pontuação "PQD1\_AB", "PQD1\_A" e "PQD1". Na primeira célula, insira o nome do professor; na segunda, o ano correspondente. Nas colunas "A1" a "C", insira a quantidade de publicações em cada categoria; na coluna "Total", faça a soma das publicações de cada ano. Nas colunas de pontuação ("PQD1\_AB", "PQD1\_A" e "PQD1"), insira os valores conforme as regras de cálculo (já valei para você). A última linha deve somar as publicações e pontuações ao longo dos anos para obter os totais finais. Segue a imagem anexa de como eu quero.



from flask import Flask, render\_template\_string, request

import pandas as pd

import zipfile

import os

import xml.etree.ElementTree as ET

app = Flask(\_\_name\_\_)

# Função para descompactar o ZIP e processar o XML

def processar\_zip(xml\_zip\_path):

try:

with zipfile.ZipFile(xml\_zip\_path, 'r') as zip\_ref:

zip\_ref.extractall('./temp\_xml')

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

if not xml\_files:

return None

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

return xml\_path

except Exception as e:

print(f"Erro ao descompactar o arquivo ZIP: {e}")

return None

# Função para extrair DOIs

def extrair\_doi(xml\_string):

root = ET.fromstring(xml\_string)

artigos\_publicados = root.findall(".//ARTIGO-PUBLICADO")

dois = []

doi\_map = {

"Estimando a geração de resíduos sólidos domiciliares a partir do consumo de água em edifícios multifamiliares.": "10.1234/exampledoi1",

"Sobre a geração de resíduos sólidos domiciliares em bairros de classe média e alta de João Pessoa": "10.1234/exampledoi2"

}

for artigo in artigos\_publicados:

dados\_basicos = artigo.find("DADOS-BASICOS-DO-ARTIGO")

titulo = dados\_basicos.get("TITULO-DO-ARTIGO", "")

doi = dados\_basicos.get("DOI", None)

if not doi and titulo in doi\_map:

doi = doi\_map[titulo]

if doi:

dois.append(doi)

return dois

# Função para processar os arquivos XML

def processar\_xml(xml\_path):

try:

tree = ET.parse(xml\_path)

root = tree.getroot()

artigos = []

with open(xml\_path, 'r', encoding='ISO-8859-1') as file:

xml\_data = file.read()

dois = extrair\_doi(xml\_data)

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

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

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

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

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

issn\_formatado = f"{issn[:4]}-{issn[4:]}" if len(issn) == 8 else issn

doi = dois[i] if i < len(dois) else 'Sem DOI'

artigos.append({'Título': titulo, 'Autores': autores, 'ISSN': issn\_formatado, 'Ano': int(ano), 'DOI': doi, 'Qualis': 'Sem Qualis'})

return pd.DataFrame(artigos)

except Exception as e:

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

return pd.DataFrame()

# Função para carregar a tabela Qualis

def carregar\_qualis\_csv(caminho\_csv):

try:

qualis\_df = pd.read\_csv(caminho\_csv, delimiter=',', encoding='latin-1', on\_bad\_lines='warn')

return qualis\_df

except Exception as e:

print(f"Erro ao carregar o arquivo CSV: {e}")

return None

# Função para associar Qualis ao artigo pelo ISSN

def associar\_qualis(df, qualis\_df):

qualis\_df = qualis\_df.drop\_duplicates(subset='ISSN')

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

df['ISSN\_normalizado'] = df['ISSN'].str.replace('-', '')

df = pd.merge(df, qualis\_df[['ISSN\_normalizado', 'Estrato']], on='ISSN\_normalizado', how='left')

df['Qualis'] = df['Estrato'].fillna('Sem Qualis')

df.drop(columns=['ISSN\_normalizado'], inplace=True)

return df

# Função para identificar DP e DISC

def identificar\_dp\_disc(artigos\_df, docentes\_df, discentes\_df):

docentes\_permanentes = docentes\_df[docentes\_df['CATEGORIA'] == 'Permanente']['NOME'].str.lower().tolist()

discentes = discentes\_df['Nome Discente'].str.lower().tolist()

def contar\_participantes(autores, lista\_nomes):

return sum(1 for autor in autores.split(', ') if any(nome in autor.lower() for nome in lista\_nomes))

artigos\_df['Num\_DP'] = artigos\_df['Autores'].apply(lambda autores: contar\_participantes(autores, docentes\_permanentes))

artigos\_df['Num\_DISC'] = artigos\_df['Autores'].apply(lambda autores: contar\_participantes(autores, discentes))

return artigos\_df

# Função para gerar a tabela final agregada

def gerar\_tabela\_resumo\_por\_ano(df):

tabela\_agrupada = df.groupby(['Ano']).agg(

A1=('A1', 'sum'),

A2=('A2', 'sum'),

A3=('A3', 'sum'),

A4=('A4', 'sum'),

B1=('B1', 'sum'),

B2=('B2', 'sum'),

B3=('B3', 'sum'),

B4=('B4', 'sum'),

C=('C', 'sum'),

Total=('Total', 'sum'),

PQD1\_AB=('PQD1\_AB', 'sum'),

PQD1\_A=('PQD1\_A', 'sum'),

PQD1=('PQD1', 'sum')

).reset\_index()

total\_row = tabela\_agrupada.sum(numeric\_only=True).to\_frame().T

total\_row['Ano'] = 'TOTAL'

tabela\_final = pd.concat([tabela\_agrupada, total\_row], ignore\_index=True)

tabela\_final['Docente'] = 'Antonio Acacio'

cols = ['Docente'] + [col for col in tabela\_final.columns if col != 'Docente']

tabela\_final = tabela\_final[cols]

return tabela\_final

# Passo 1 - Interface de Upload

@app.route('/', methods=['GET', 'POST'])

def home():

if request.method == 'POST':

xml\_zip\_file = request.files['file']

xml\_zip\_path = f"./{xml\_zip\_file.filename}"

xml\_zip\_file.save(xml\_zip\_path)

xml\_path = processar\_zip(xml\_zip\_path)

if not xml\_path:

return "Erro ao descompactar o ZIP ou encontrar o XML."

ano\_inicio = int(request.form.get('ano\_inicio'))

ano\_fim = int(request.form.get('ano\_fim'))

df = processar\_xml(xml\_path)

qualis\_df = carregar\_qualis\_csv(r'C:\Users\Decivil\Dropbox\PPGEC\Indicadores\chatgpt\flask\_app\Classificação Qualis 2017\_2020.csv')

df = associar\_qualis(df, qualis\_df)

df = df[(df['Ano'] >= ano\_inicio) & (df['Ano'] <= ano\_fim)]

docentes\_df = pd.read\_excel(r'C:\Users\Decivil\Dropbox\PPGEC\Indicadores\chatgpt\flask\_app\Docentes PPGEC.xlsx')

discentes\_df = pd.read\_csv(r'C:\Users\Decivil\Dropbox\PPGEC\Indicadores\chatgpt\flask\_app\Alunos 2021 - Discentes.csv')

df = identificar\_dp\_disc(df, docentes\_df, discentes\_df)

# Exibir interface de ajuste de Num\_DP e Num\_DISC

df['Num\_DP'] = df['Num\_DP'].apply(lambda x: f'<input type="text" name="Num\_DP" value="{x}">')

df['Num\_DISC'] = df['Num\_DISC'].apply(lambda x: f'<input type="text" name="Num\_DISC" value="{x}">')

html\_output = df.to\_html(index=False, escape=False)

html = '''

<html>

<head>

<title>PPGEC - Sistema de Pontuação</title>

</head>

<body>

<h1>PPGEC - Resultados de Artigos (Editável)</h1>

<form method="POST" action="/submit">

''' + html\_output + '''

<input type="submit" value="Salvar Alterações">

</form>

<br><br>

<a href="/">Voltar</a>

</body>

</html>

'''

return render\_template\_string(html)

# Interface de upload de ZIP e parâmetros de ano (Passo 1)

html = '''

<html>

<head>

<title>PPGEC - Sistema de Pontuação</title>

</head>

<body>

<h1>PPGEC - Análise de Artigos - Passo 1</h1>

<form method="POST" enctype="multipart/form-data">

<label for="file">Carregar ZIP:</label>

<input type="file" name="file" required><br><br>

<label for="ano\_inicio">Ano de Início:</label>

<input type="text" name="ano\_inicio" required><br><br>

<label for="ano\_fim">Ano de Fim:</label>

<input type="text" name="ano\_fim" required><br><br>

<input type="submit" value="Processar">

</form>

</body>

</html>

'''

return render\_template\_string(html)

# Passo 2 - Interface para ajustar Num\_DP, Num\_DISC, Ponderação

# Função para adicionar a coluna DOI com link e seta verde

def adicionar\_coluna\_doi(df):

df['DOI'] = df['DOI'].apply(lambda doi: f'<a href="https://doi.org/{doi}" target="\_blank">🡆</a>' if doi != 'Sem DOI' else 'Sem DOI')

return df

# Função para gerar a tabela com as colunas na ordem correta

def gerar\_tabela\_editavel(df):

# Adiciona a coluna DOI com a seta verde

df = adicionar\_coluna\_doi(df)

# Coloca os campos editáveis para Num\_DP, Num\_DISC e Ponderação Discente

df['Num\_DP'] = df['Num\_DP'].apply(lambda x: f'<input type="text" name="Num\_DP" value="{x}">')

df['Num\_DISC'] = df['Num\_DISC'].apply(lambda x: f'<input type="text" name="Num\_DISC" value="{x}">')

df['Ponderação Discente'] = df['Ponderação Discente'].apply(lambda x: f'<input type="text" name="Ponderacao\_Discente" value="{x}">')

# Reorganizar as colunas na ordem correta

df = df[['Título', 'Autores', 'ISSN', 'Ano', 'DOI', 'Qualis', 'Num\_DP', 'Num\_DISC', 'Ponderação Qualis', 'Ponderação Discente']]

# Gera o HTML da tabela

html\_output = df.to\_html(index=False, escape=False)

return html\_output

@app.route('/submit', methods=['POST'])

def submit():

# Captura os valores enviados pelo usuário para Num\_DP, Num\_DISC e Ponderação Discente

num\_dp = request.form.getlist('Num\_DP')

num\_disc = request.form.getlist('Num\_DISC')

ponderacao\_discente = request.form.getlist('Ponderacao\_Discente')

# Processar o DataFrame com os valores editados

# Neste ponto você pode salvar ou processar os valores conforme necessário

# Aqui estou apenas simulando a próxima etapa

return '''

<html>

<head>

<title>PPGEC - Tabela Final</title>

</head>

<body>

<h1>Alterações salvas! Clique para gerar a tabela final.</h1>

<form method="POST" action="/tabela\_resumo">

<input type="submit" value="Gerar Tabela Final">

</form>

</body>

</html>

'''

# Passo 2 - Interface para ajustar Num\_DP, Num\_DISC, Ponderação

@app.route('/passo2', methods=['POST'])

def passo2():

# Carregar e processar os dados

xml\_zip\_file = request.files['file']

xml\_zip\_path = f"./{xml\_zip\_file.filename}"

xml\_zip\_file.save(xml\_zip\_path)

xml\_path = processar\_zip(xml\_zip\_path)

if not xml\_path:

return "Erro ao descompactar o ZIP ou encontrar o XML."

ano\_inicio = int(request.form.get('ano\_inicio'))

ano\_fim = int(request.form.get('ano\_fim'))

df = processar\_xml(xml\_path)

qualis\_df = carregar\_qualis\_csv(r'C:\Users\Decivil\Dropbox\PPGEC\Indicadores\chatgpt\flask\_app\Classificação Qualis 2017\_2020.csv')

df = associar\_qualis(df, qualis\_df)

df = df[(df['Ano'] >= ano\_inicio) & (df['Ano'] <= ano\_fim)]

docentes\_df = pd.read\_excel(r'C:\Users\Decivil\Dropbox\PPGEC\Indicadores\chatgpt\flask\_app\Docentes PPGEC.xlsx')

discentes\_df = pd.read\_csv(r'C:\Users\Decivil\Dropbox\PPGEC\Indicadores\chatgpt\flask\_app\Alunos 2021 - Discentes.csv')

df = identificar\_dp\_disc(df, docentes\_df, discentes\_df)

# Aplicar ponderações padrão (sem edição aqui)

qualis\_ponderacao = {

'A1': 1.00, 'A2': 0.90, 'A3': 0.75, 'A4': 0.60,

'B1': 0.40, 'B2': 0.30, 'B3': 0.15, 'B4': 0.05, 'C': 0

}

ponderacao\_discente\_padrao = 0.75

df = calcular\_ponderacao(df, qualis\_ponderacao, ponderacao\_discente\_padrao)

# Gera a tabela para a interface de ajuste

html\_output = gerar\_tabela\_editavel(df)

# Exibir a tabela com os campos editáveis

html = '''

<html>

<head>

<title>PPGEC - Ajuste de Artigos - Passo 2</title>

</head>

<body>

<h1>PPGEC - Ajuste de Artigos (Passo 2)</h1>

<form method="POST" action="/submit">

''' + html\_output + '''

<input type="submit" value="Salvar Alterações">

</form>

<br><br>

<a href="/">Voltar</a>

</body>

</html>

'''

return render\_template\_string(html)

# Passo 3 - Geração da Tabela Final (Pontuação)

@app.route('/tabela\_resumo', methods=['POST'])

def gerar\_tabela\_resumo():

# Aqui, as informações da tabela final serão geradas com base nos dados

# Resumo da tabela por ano, docente, etc.

tabela\_final = gerar\_tabela\_resumo\_por\_ano(df) # Função que gera a tabela final

# Exibir a tabela no formato final

html\_output = tabela\_final.to\_html(index=False, escape=False)

html = '''

<html>

<head>

<title>Tabela Final de Pontuação</title>

</head>

<body>

<h1>Tabela Final de Pontuação por Docente e Ano</h1>

''' + html\_output + '''

<br><br>

<a href="/">Voltar</a>

</body>

</html>

'''

return render\_template\_string(html)

if \_\_name\_\_ == '\_\_main\_\_':

if not os.path.exists('./temp\_xml'):

os.makedirs('./temp\_xml')

app.run(debug=True)