



Erro popup

COMEÇA A CONFERIR PORÉ LOGO EM SEGUIDA VEM O ERRO “Ocorreu um erro ao proessar os jogos”

Erro detalhado:

Object { mensagem: "Erro ao processar jogos", pilha: "@http://127.0.0.1:5000/static/js/main.js:1282:41\n", status: undefined, resposta: undefined }

Erro console navagedor

2025-01-26 10:14:23,887 - Mega Sena Conferidor - ERROR - Erro na conferência: 0

Erro no visual code terminal

Estrutura

Mode LastWriteTime Length Name

---- ------------- ------ ----

d----- 19/01/2025 15:16 Requisitos

d----- 19/01/2025 15:16 static

d----- 24/01/2025 15:06 templates

d----- 19/01/2025 17:02 venv

d----- 22/01/2025 08:23 .vscode

d----- 24/01/2025 15:41 \_\_pycache\_\_

------ 24/01/2025 14:40 1075 LICENSE

------ 24/01/2025 14:40 3957 MegaSenaConferidor.ps1

------ 24/01/2025 14:40 1650 README.md

------ 25/01/2025 09:03 18567 app.py

------ 25/01/2025 07:52 277 requirements.txt

------ 24/01/2025 14:40 1745 setup.ps1

------ 25/01/2025 09:05 5 .gitignore

------ 25/01/2025 09:12 3120 redis\_config.py

------ 25/01/2025 09:04 165 .env

# requirements.txt

# flask==2.0.3 novo

flask[async]

Werkzeug==2.0.3

requests==2.26.0

python-dotenv==0.19.0

aiohttp --only-binary=:all: # Para requisições assíncronas novo

asyncio==3.4.3 # Para processar múltiplas requisições novo

redis # novo

pandas # novo

python-dotenv # novo

asyncio # novo

waitress # novo

import redis

import os

from typing import Optional

import logging

import json

from ast import literal\_eval

from dotenv import load\_dotenv

# Carrega as variáveis do arquivo .env

load\_dotenv()

class RedisConfig:

def \_\_init\_\_(self):

self.logger = self.\_setup\_logger()

self.redis\_client = self.\_initialize\_redis()

self.CACHE\_EXPIRATION = 60 \* 60 \* 24 # 24 horas

def \_setup\_logger(self) -> logging.Logger:

logger = logging.getLogger('RedisConfig')

logger.setLevel(logging.INFO)

handler = logging.StreamHandler()

formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')

handler.setFormatter(formatter)

logger.addHandler(handler)

return logger

def \_initialize\_redis(self) -> Optional[redis.Redis]:

try:

redis\_url = os.getenv("REDIS\_URL")

if redis\_url: # Prioridade para a URL completa, se definida

client = redis.from\_url(redis\_url)

client.ping()

self.logger.info("Connected to Redis via URL")

return client

else:

# Conexão usando host, port e senha individuais

client = redis.Redis(

host=os.getenv('REDIS\_HOST'),

port=int(os.getenv('REDIS\_PORT')),

password=os.getenv('REDIS\_PASSWORD'),

db=int(os.getenv('REDIS\_DB', 0)),

decode\_responses=True,

socket\_timeout=5,

retry\_on\_timeout=True

)

client.ping()

self.logger.info("Connected to Redis via parameters")

return client

except Exception as e:

self.logger.error(f"A conexão com o Redis falhou: {str(e)}")

return None

def get\_cached\_result(self, concurso: int) -> Optional[dict]:

if not self.redis\_client:

return None

try:

cached = self.redis\_client.get(f"megasena:{concurso}")

if not cached:

return None

try:

return json.loads(cached)

except:

try:

return literal\_eval(cached)

except:

return None

except Exception as e:

self.logger.error(f"Cache retrieval error for contest {concurso}: {str(e)}")

return None

def set\_cached\_result(self, concurso: int, data: dict) -> bool:

if not self.redis\_client:

return False

try:

serialized = json.dumps(data, ensure\_ascii=False)

self.redis\_client.setex(

f"megasena:{concurso}",

self.CACHE\_EXPIRATION,

serialized

)

return True

except Exception as e:

self.logger.error(f"Cache storage error for contest {concurso}: {str(e)}")

return False

redis\_config = RedisConfig()

from flask import Flask, render\_template, request, jsonify, send\_file

from redis\_config import redis\_config

from datetime import datetime

from ast import literal\_eval

from dotenv import load\_dotenv

from typing import Optional

import requests

import os

import random

import aiohttp

import asyncio

import redis

import json

import pandas as pd

import io

import logging

# Carrega as variáveis do arquivo .env

load\_dotenv()

class RedisConfig:

def \_\_init\_\_(self):

self.logger = self.\_setup\_logger()

self.redis\_client = self.\_initialize\_redis()

self.CACHE\_EXPIRATION = 60 \* 60 \* 24 # 24 horas

def \_setup\_logger(self) -> logging.Logger:

logger = logging.getLogger('RedisConfig')

logger.setLevel(logging.INFO)

handler = logging.StreamHandler()

formatter = logging.Formatter('%(asctime)s - %(name)s - %(levelname)s - %(message)s')

handler.setFormatter(formatter)

logger.addHandler(handler)

return logger

def \_initialize\_redis(self) -> Optional[redis.Redis]:

try:

redis\_url = os.getenv("REDIS\_URL")

if redis\_url: # Prioridade para a URL completa, se definida

client = redis.from\_url(redis\_url)

client.ping()

self.logger.info("Connected to Redis via URL")

return client

else:

# Conexão usando host, port e senha individuais

client = redis.Redis(

host=os.getenv('REDIS\_HOST'),

port=int(os.getenv('REDIS\_PORT')),

password=os.getenv('REDIS\_PASSWORD'),

db=int(os.getenv('REDIS\_DB', 0)),

decode\_responses=True,

socket\_timeout=5,

retry\_on\_timeout=True

)

client.ping()

self.logger.info("Connected to Redis via parameters")

return client

except Exception as e:

self.logger.error(f"Redis connection failed: {str(e)}")

return None

def get\_cached\_result(self, concurso: int) -> Optional[dict]:

if not self.redis\_client:

return None

try:

cached = self.redis\_client.get(f"megasena:{concurso}")

if not cached:

return None

try:

return json.loads(cached)

except:

try:

return literal\_eval(cached)

except:

return None

except Exception as e:

self.logger.error(f"Cache retrieval error for contest {concurso}: {str(e)}")

return None

def set\_cached\_result(self, concurso: int, data: dict) -> bool:

if not self.redis\_client:

return False

try:

serialized = json.dumps(data, ensure\_ascii=False)

self.redis\_client.setex(

f"megasena:{concurso}",

self.CACHE\_EXPIRATION,

serialized

)

return True

except Exception as e:

self.logger.error(f"Cache storage error for contest {concurso}: {str(e)}")

return False

redis\_config = RedisConfig()

# Configuração básica de logs

logging.basicConfig(level=logging.INFO, format='%(asctime)s - %(name)s - %(levelname)s - %(message)s')

# Uso de logger

logger = logging.getLogger("Mega Sena Conferidor")

logger.info("Mensagem de log de teste")

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

def get\_cached\_result(concurso):

return redis\_config.get\_cached\_result(concurso)

def set\_cached\_result(concurso, data):

return redis\_config.set\_cached\_result(concurso, data)

def atualizar\_estatisticas\_jogo(stats, jogo, acertos):

jogo\_key = tuple(sorted(jogo))

if jogo\_key not in stats:

stats[jogo\_key] = {

'total': 0,

'distribuicao': {1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0},

'numeros': list(jogo\_key)

}

stats[jogo\_key]['total'] += 1

stats[jogo\_key]['distribuicao'][acertos] += 1

return stats

API\_BASE\_URL = "https://loteriascaixa-api.herokuapp.com/api"

async def get\_latest\_result():

try:

async with aiohttp.ClientSession() as session:

async with session.get(f"{API\_BASE\_URL}/megasena/latest") as response:

if response.status == 200:

return await response.json()

return None

except Exception as e:

logger.info(f"Erro ao buscar último resultado: {str(e)}")

return None

@app.route('/')

async def index():

latest = await get\_latest\_result()

ultimo\_concurso = latest['concurso'] if latest else 2680

return render\_template('index.html', ultimo\_concurso=ultimo\_concurso)

@app.route('/gerar\_numeros')

async def gerar\_numeros():

numeros = random.sample(range(1, 61), 6)

return jsonify({'numeros': sorted(numeros)})

async def fetch\_concurso\_result(concurso):

try:

response = await aiohttp.ClientSession().get(f"{API\_BASE\_URL}/megasena/{concurso}")

if response.status == 200:

return await response.json()

except Exception as e:

logger.info(f"Erro ao buscar resultado do concurso {concurso}: {str(e)}")

return None

async def processar\_chunk\_jogos(session, chunk\_jogos, concurso\_results, resultados, jogos\_stats):

for resultado in concurso\_results:

if not resultado:

continue

dezenas = [int(d) for d in resultado['dezenas']]

for jogo in chunk\_jogos:

acertos = len(set(jogo) & set(dezenas))

if acertos > 0:

jogos\_stats = atualizar\_estatisticas\_jogo(jogos\_stats, jogo, acertos)

if acertos >= 4:

premio = 0

for premiacao in resultado['premiacoes']:

if ((acertos == 6 and premiacao['descricao'] == '6 acertos') or

(acertos == 5 and premiacao['descricao'] == '5 acertos') or

(acertos == 4 and premiacao['descricao'] == '4 acertos')):

premio = premiacao['valorPremio']

resultados['resumo']['total\_premios'] += premio

break

if acertos == 4: resultados['resumo']['quatro'] += 1

elif acertos == 5: resultados['resumo']['cinco'] += 1

elif acertos == 6: resultados['resumo']['seis'] += 1

resultados['acertos'].append({

'concurso': resultado['concurso'],

'data': resultado['data'],

'local': resultado.get('local', ''),

'numeros\_sorteados': dezenas,

'seus\_numeros': jogo,

'acertos': acertos,

'premio': premio

})

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

# async def conferir():

# try:

# data = await request.get\_json()

# inicio = int(data['inicio'])

# fim = int(data['fim'])

# jogos = data['jogos']

# CHUNK\_SIZE = 5000 # Processamento em chunks menores

# resultados = {

# 'acertos': [],

# 'resumo': {'quatro': 0, 'cinco': 0, 'seis': 0, 'total\_premios': 0}

# }

# jogos\_stats = {}

# # Busca resultados dos concursos uma única vez

# async with aiohttp.ClientSession() as session:

# tasks = []

# for concurso in range(inicio, fim + 1):

# cached = get\_cached\_result(concurso)

# if cached:

# tasks.append(cached)

# else:

# tasks.append(fetch\_concurso\_result(concurso))

# concursos\_results = await asyncio.gather(\*tasks)

# # Processa jogos em chunks

# for i in range(0, len(jogos), CHUNK\_SIZE):

# chunk = jogos[i:i + CHUNK\_SIZE]

# await processar\_chunk\_jogos(session, chunk, concursos\_results, resultados, jogos\_stats)

# resultados['resumo']['total\_premios'] = round(resultados['resumo']['total\_premios'], 2)

# jogos\_stats\_ordenados = sorted(

# [{'numeros': stats['numeros'],

# 'total': stats['total'],

# 'distribuicao': stats['distribuicao']}

# for stats in jogos\_stats.values()],

# key=lambda x: x['total'],

# reverse=True

# )[:10]

# return jsonify({

# 'acertos': resultados['acertos'],

# 'resumo': resultados['resumo'],

# 'jogos\_stats': jogos\_stats\_ordenados

# })

# except Exception as e:

# logger.error(f"Erro na conferência: {str(e)}")

# return jsonify({'error': str(e)}), 500

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

async def conferir():

try:

data = request.get\_json() # Não precisa de await aqui

inicio = int(data['inicio'])

fim = int(data['fim'])

jogos = data['jogos']

CHUNK\_SIZE = 5000

resultados = {

'acertos': [],

'resumo': {'quatro': 0, 'cinco': 0, 'seis': 0, 'total\_premios': 0},

'jogos\_stats': {}

}

async with aiohttp.ClientSession() as session:

tasks = []

for concurso in range(inicio, fim + 1):

url = f"{API\_BASE\_URL}/megasena/{concurso}"

tasks.append(session.get(url))

respostas = await asyncio.gather(\*tasks)

concursos\_results = []

for resp in respostas:

if resp.status == 200:

concursos\_results.append(await resp.json())

else:

concursos\_results.append(None)

for resultado in concursos\_results:

if not resultado:

continue

dezenas = [int(d) for d in resultado['dezenas']]

for jogo in jogos:

acertos = len(set(jogo) & set(dezenas))

if acertos >= 4:

premio = 0

for premiacao in resultado['premiacoes']:

if ((acertos == 6 and premiacao['descricao'] == '6 acertos') or

(acertos == 5 and premiacao['descricao'] == '5 acertos') or

(acertos == 4 and premiacao['descricao'] == '4 acertos')):

premio = premiacao['valorPremio']

resultados['resumo']['total\_premios'] += premio

break

if acertos == 4: resultados['resumo']['quatro'] += 1

elif acertos == 5: resultados['resumo']['cinco'] += 1

elif acertos == 6: resultados['resumo']['seis'] += 1

resultados['acertos'].append({

'concurso': resultado['concurso'],

'data': resultado['data'],

'local': resultado.get('local', ''),

'numeros\_sorteados': dezenas,

'seus\_numeros': jogo,

'acertos': acertos,

'premio': premio

})

jogo\_key = tuple(sorted(jogo))

if jogo\_key not in resultados['jogos\_stats']:

resultados['jogos\_stats'][jogo\_key] = {

'numeros': list(jogo\_key),

'total': 0,

'distribuicao': {1: 0, 2: 0, 3: 0, 4: 0, 5: 0, 6: 0}

}

resultados['jogos\_stats'][jogo\_key]['total'] += 1

resultados['jogos\_stats'][jogo\_key]['distribuicao'][acertos] += 1

resultados['jogos\_stats'] = sorted(

resultados['jogos\_stats'].values(),

key=lambda x: x['total'],

reverse=True

)[:10]

return jsonify(resultados)

except Exception as e:

logger.error(f"Erro na conferência: {str(e)}")

return jsonify({'error': str(e)}), 500

@app.route('/processar\_arquivo', methods=['POST'])

def processar\_arquivo():

if 'file' not in request.files:

return jsonify({'error': 'Nenhum arquivo enviado'}), 400

file = request.files['file']

if file.filename == '':

return jsonify({'error': 'Nenhum arquivo selecionado'}), 400

try:

jogos = []

# Tenta processar como texto primeiro

if file.filename.endswith('.txt'):

try:

content = file.read().decode('utf-8-sig')

for line in content.strip().split('\n'):

try:

line = ''.join(c for c in line if c.isdigit() or c.isspace())

numbers = [int(n) for n in line.strip().split() if n.strip()]

if (len(numbers) == 6 and

all(1 <= n <= 60 for n in numbers) and

len(set(numbers)) == 6):

jogos.append(sorted(numbers))

except (ValueError, TypeError):

continue

except Exception as txt\_error:

logger.info(f"Erro ao processar TXT: {str(txt\_error)}")

# Se não é txt, tenta processar como Excel

elif file.filename.endswith(('.xlsx', '.xls')):

try:

content = file.read()

df = pd.read\_excel(io.BytesIO(content))

for \_, row in df.iterrows():

numbers = []

for val in row.values[:6]:

try:

num = int(float(val))

if 1 <= num <= 60:

numbers.append(num)

except (ValueError, TypeError):

continue

if len(numbers) == 6 and len(set(numbers)) == 6:

jogos.append(sorted(numbers))

except Exception as excel\_error:

logger.info(f"Erro ao processar Excel: {str(excel\_error)}")

else:

return jsonify({

'error': 'Formato de arquivo não suportado. Use .txt ou .xlsx'

}), 400

if not jogos:

return jsonify({

'error': 'Nenhum jogo válido encontrado no arquivo'

}), 400

return jsonify({'jogos': jogos})

except UnicodeDecodeError:

return jsonify({

'error': 'Erro ao ler o arquivo. Certifique-se que é um arquivo de texto válido.'

}), 400

except Exception as e:

logger.info(f"Erro ao processar arquivo: {str(e)}")

return jsonify({

'error': 'Erro ao processar o arquivo. Verifique o formato e tente novamente.'

}), 500

@app.route('/exportar/<tipo>/<formato>', methods=['POST'])

def exportar\_dados(tipo, formato):

try:

data = request.json

if tipo == 'resumo-acertos':

return exportar\_resumo\_acertos(data, formato)

elif tipo == 'jogos-premiados':

return exportar\_jogos\_premiados(data, formato)

elif tipo == 'jogos-sorteados':

return exportar\_jogos\_sorteados(data, formato)

else:

return jsonify({'error': 'Tipo de exportação inválido'}), 400

except Exception as e:

logger.info(f"Erro na exportação: {str(e)}")

return jsonify({'error': str(e)}), 500

def exportar\_resumo\_acertos(data, formato):

try:

# Criação do DataFrame com os dados fornecidos

df = pd.DataFrame({

'Tipo': ['4 Acertos', '5 Acertos', '6 Acertos', 'Total de Prêmios'],

'Quantidade': [

data['resumo']['quatro'],

data['resumo']['cinco'],

data['resumo']['seis'],

''

],

'Valor Total': [

data['resumo'].get('valor\_quatro', 0),

data['resumo'].get('valor\_cinco', 0),

data['resumo'].get('valor\_seis', 0),

data['resumo'].get('total\_premios', 0)

]

})

# Exportação para formato Excel

if formato == 'xlsx':

output = io.BytesIO()

with pd.ExcelWriter(output, engine='xlsxwriter') as writer:

df.to\_excel(writer, sheet\_name='Resumo de Acertos', index=False)

output.seek(0)

return send\_file(

output,

mimetype='application/vnd.openxmlformats-officedocument.spreadsheetml.sheet',

as\_attachment=True,

download\_name='resumo\_acertos.xlsx'

)

# Exportação para formato HTML

elif formato == 'html':

html\_content = df.to\_html(index=False, classes='table table-striped')

html = f'''

<html>

<head>

<style>

table {{ border-collapse: collapse; width: 100%; }}

th, td {{ border: 1px solid black; padding: 8px; text-align: left; }}

th {{ background-color: #008751; color: white; }}

</style>

</head>

<body>

<h2>Resumo de Acertos</h2>

{html\_content}

</body>

</html>

'''

return send\_file(

io.BytesIO(html.encode()),

mimetype='text/html',

as\_attachment=True,

download\_name='resumo\_acertos.html'

)

# Caso o formato não seja suportado

else:

raise ValueError(f"Formato '{formato}' não suportado. Use 'xlsx' ou 'html'.")

except Exception as e:

raise Exception(f"Erro ao exportar resumo: {str(e)}")

def exportar\_jogos\_premiados(data, formato):

try:

rows = []

for resultado in data['acertos']:

rows.append({

'Concurso': resultado['concurso'],

'Data': resultado['data'],

'Local': resultado['local'],

'Números Sorteados': ' '.join(str(n).zfill(2) for n in sorted(resultado['numeros\_sorteados'])),

'Seu Jogo': ' '.join(str(n).zfill(2) for n in sorted(resultado['seus\_numeros'])),

'Acertos': resultado['acertos'],

'Prêmio': f"R$ {resultado['premio']:,.2f}",

'Status': 'Premiado' if resultado['premio'] > 0 else 'Acumulado'

})

df = pd.DataFrame(rows)

if formato == 'xlsx':

output = io.BytesIO()

with pd.ExcelWriter(output, engine='xlsxwriter') as writer:

df.to\_excel(writer, sheet\_name='Jogos Premiados', index=False)

output.seek(0)

return send\_file(

output,

mimetype='application/vnd.openxmlformats-officedocument.spreadsheetml.sheet',

as\_attachment=True,

download\_name='jogos\_premiados.xlsx'

)

elif formato == 'html':

html\_content = df.to\_html(index=False, classes='table table-striped')

html = f'''

<html>

<head>

<style>

table {{ border-collapse: collapse; width: 100%; }}

th, td {{ border: 1px solid black; padding: 8px; text-align: left; }}

th {{ background-color: #008751; color: white; }}

</style>

</head>

<body>

<h2>Jogos Premiados</h2>

{html\_content}

</body>

</html>

'''

return send\_file(

io.BytesIO(html.encode()),

mimetype='text/html',

as\_attachment=True,

download\_name='jogos\_premiados.html'

)

except Exception as e:

raise Exception(f"Erro ao exportar jogos premiados: {str(e)}")

def exportar\_jogos\_sorteados(data, formato):

try:

rows = []

for jogo in data['jogos\_stats']:

distribuicao = []

for i in range(1, 7):

if jogo['distribuicao'].get(i, 0) > 0:

distribuicao.append(f"{i} pontos: {jogo['distribuicao'][i]}")

rows.append({

'Jogo': ' '.join(str(n).zfill(2) for n in jogo['numeros']),

'Total': jogo['total'],

'Distribuição': ', '.join(distribuicao)

})

df = pd.DataFrame(rows)

if formato == 'xlsx':

output = io.BytesIO()

with pd.ExcelWriter(output, engine='xlsxwriter') as writer:

df.to\_excel(writer, sheet\_name='Jogos Mais Sorteados', index=False)

output.seek(0)

return send\_file(

output,

mimetype='application/vnd.openxmlformats-officedocument.spreadsheetml.sheet',

as\_attachment=True,

download\_name='jogos\_sorteados.xlsx'

)

elif formato == 'html':

html\_content = df.to\_html(index=False, classes='table table-striped')

html = f'''

<html>

<head>

<meta charset="utf-8">

<style>

table {{ border-collapse: collapse; width: 100%; }}

th, td {{ border: 1px solid black; padding: 8px; text-align: left; }}

th {{ background-color: #008751; color: white; }}

tr:nth-child(even) {{ background-color: #f2f2f2; }}

tr:hover {{ background-color: #ddd; }}

</style>

</head>

<body>

<h2>Jogos Mais Sorteados</h2>

{html\_content}

</body>

</html>

'''

return send\_file(

io.BytesIO(html.encode('utf-8')),

mimetype='text/html',

as\_attachment=True,

download\_name='jogos\_sorteados.html'

)

except Exception as e:

raise Exception(f"Erro ao exportar jogos sorteados: {str(e)}")

def exportar\_jogos\_sorteados(data, formato):

try:

# Prepara as linhas de dados

rows = []

for jogo in data['jogos\_stats']:

distribuicao = []

# Monta a distribuição de acertos (1 a 6 pontos)

for i in range(1, 7):

if jogo['distribuicao'].get(i, 0) > 0:

distribuicao.append(f"{i} pontos: {jogo['distribuicao'][i]}")

# Adiciona cada jogo com suas estatísticas

rows.append({

'Jogo': ' '.join(str(n).zfill(2) for n in jogo['numeros']),

'Total': jogo['total'],

'Distribuição': ', '.join(distribuicao)

})

# Cria DataFrame com os dados

df = pd.DataFrame(rows)

# Exportação para Excel

if formato == 'xlsx':

output = io.BytesIO()

with pd.ExcelWriter(output, engine='xlsxwriter') as writer:

df.to\_excel(writer, sheet\_name='Jogos Mais Sorteados', index=False)

output.seek(0)

return send\_file(

output,

mimetype='application/vnd.openxmlformats-officedocument.spreadsheetml.sheet',

as\_attachment=True,

download\_name='jogos\_sorteados.xlsx'

)

# Exportação para HTML

elif formato == 'html':

html\_content = df.to\_html(index=False, classes='table table-striped')

html = f'''

<html>

<head>

<meta charset="utf-8">

<style>

table {{ border-collapse: collapse; width: 100%; }}

th, td {{ border: 1px solid black; padding: 8px; text-align: left; }}

th {{ background-color: #008751; color: white; }}

tr:nth-child(even) {{ background-color: #f2f2f2; }}

tr:hover {{ background-color: #ddd; }}

</style>

</head>

<body>

<h2>Jogos Mais Sorteados</h2>

{html\_content}

</body>

</html>

'''

return send\_file(

io.BytesIO(html.encode('utf-8')),

mimetype='text/html',

as\_attachment=True,

download\_name='jogos\_sorteados.html'

)

except Exception as e:

raise Exception(f"Erro ao exportar jogos sorteados: {str(e)}")

"""

#PARA O LOCALHOST

# timeout do servidor para evitar desconexões:

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

app.run(debug=True, host='0.0.0.0', port=5000)

"""

#PARA O SERVIDOR

# timeout do servidor para evitar desconexões:

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

app.run(debug=False, host='0.0.0.0', port=5000)

"""

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

app.run(debug=False, host='0.0.0.0', port=5000)

# LOCALHOST

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

app.run(debug=True)

# Agora a parte de configuração da porta

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

port = int(os.environ.get("PORT", 5000)) # Obtém a porta do ambiente ou usa 5000 como padrão

app.run(host="0.0.0.0", port=port) # Inicia o servidor Flask na porta correta

"""

#exportar\_dados

#exportar\_resumo\_acertos

#exportar\_jogos\_premiados

#exportar\_jogos\_sorteados

<!DOCTYPE html>

<html lang="pt-BR">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>MegaSena Conferidor</title>

<link rel="stylesheet" href="{{ url\_for('static', filename='css/style.css') }}">

</head>

<body>

<div class="container">

<h1>Conferidor MegaSena</h1>

<h1>Informação importante!</h1>

<h2>Para os Jogos:</h2>

<ul>

<li>Você pode carregar quantos jogos quiser (20 mil, 80 mil, 100 mil...)</li>

<li>Não há limite para quantidade de jogos</li>

<li>Todos os jogos serão verificados em cada lote</li>

</ul>

<h2>Para os Concursos:</h2>

<ul>

<li>O sistema agora processa em lotes de 930 concursos</li>

<li>

Por exemplo, se você quiser verificar do concurso 1 até o 2816:

<ul>

<li>Lote 1: concursos 1-930</li>

<li>Lote 2: concursos 931-1860</li>

<li>Lote 3: concursos 1861-2790</li>

<li>Lote 4: concursos 2791-2816</li>

</ul>

</li>

</ul>

<h2>Acumulação de Resultados:</h2>

<ul>

<li>O sistema mantém um acumulador que vai somando:</li>

<ul>

<li>Quantidade de acertos (quadra, quina, sena)</li>

<li>Valores dos prêmios</li>

<li>Todos os jogos premiados</li>

<li>Estatísticas gerais</li>

</ul>

</ul>

<p>

<pre>

NA PRÁTICA...

Se você tem 50.000 jogos:

Lote 1 (1-930):

- Processa os 50.000 jogos

- Guarda resultados

↓

Lote 2 (931-1860):

- Processa os mesmos 50.000 jogos

- Soma com resultados anteriores

↓

E assim por diante...

</pre>

Os lotes sempre serão divididos de 930 à 930, então no <b><u>Concurso Inicial</u></b> adicione o concurso que quiser, selecione o checkbox <b><u>Somente Premiados</u></b>, e aguarde a conferencia.

</p>

<div class="volante">

<h2>Selecione 6 números</h2>

<div class="numeros">

{% for i in range(1, 61) %}

<button type="button" class="numero" data-numero="{{ i }}">{{ '%02d' % i }}</button>

{% endfor %}

</div>

</div>

<div class="drop-zone" id="drop-zone">

<div class="drop-zone-content">

<p>Arraste e solte seu arquivo aqui ou clique para selecionar</p>

<p class="drop-zone-hint">Arquivos .txt ou .xlsx</p>

<input type="file" id="file-input" accept=".txt,.xlsx" hidden>

</div>

</div>

<div class="controles">

<div class="input-group">

<label for="inicio">Concurso Inicial:</label>

<input type="number" id="inicio" min="1">

</div>

<div class="input-group">

<label for="fim">Concurso Final:</label>

<input type="number" id="fim" min="1" value="{{ ultimo\_concurso }}">

</div>

<div class="checkbox-group">

<input type="checkbox" id="somente-premiados">

<label for="somente-premiados">Somente premiados</label>

</div>

</div>

<div class="acoes">

<button id="incluir" class="btn btn-azul">Incluir Jogo</button>

<button id="limpar" class="btn btn-vermelho">Limpar</button>

<button id="sugestao" class="btn btn-roxo">Palpite</button>

<button id="conferir" class="btn btn-verde">Conferir</button>

</div>

<div class="jogos-incluidos">

<h3>Jogos Incluídos (<span id="contador-jogos">0</span> jogos)</h3>

<div id="lista-jogos" class="lista-jogos"></div>

</div>

<div class="acoes-jogos">

<button id="btn-remover-selecionados" class="btn-acao" disabled>

Remover Selecionados

</button>

<button id="btn-limpar-todos" class="btn-acao">

Limpar Todos

</button>

</div>

<div id="overlay" class="overlay" style="display: none;">

<div class="progress">

<div class="progress-bar"></div>

<div class="progress-text">Conferindo jogos...</div>

<button id="btn-cancelar-conferencia" class="btn-cancelar">Cancelar Conferência</button>

</div>

</div>

<div class="resultados">

<h2>Resumo de acertos</h2>

<div class="export-buttons hidden">

<button onclick="exportarDados('resumo-acertos', 'xlsx')" class="btn-export">

<i class="fas fa-file-excel"></i> Excel

</button>

<button onclick="exportarDados('resumo-acertos', 'html')" class="btn-export">

<i class="fas fa-file-code"></i> HTML

</button>

</div>

<div class="cards">

<div class="card">

<h3>4 Acertos</h3>

<p class="contagem" id="quatro-acertos">0</p>

<p class="valor-premio" id="quatro-valor">R$ 0,00</p>

</div>

<div class="card">

<h3>5 Acertos</h3>

<p class="contagem" id="cinco-acertos">0</p>

<p class="valor-premio" id="cinco-valor">R$ 0,00</p>

</div>

<div class="card">

<h3>6 Acertos</h3>

<p class="contagem" id="seis-acertos">0</p>

<p class="valor-premio" id="seis-valor">R$ 0,00</p>

</div>

</div>

<div id="detalhes-resultados" class="detalhes-resultados"></div>

<div class="tabela-resumo">

<h2>Resumo dos Jogos Premiados</h2>

<div class="export-buttons hidden">

<button onclick="exportarDados('jogos-premiados', 'xlsx')" class="btn-export">

<i class="fas fa-file-excel"></i> Excel

</button>

<button onclick="exportarDados('jogos-premiados', 'html')" class="btn-export">

<i class="fas fa-file-code"></i> HTML

</button>

</div>

<div class="tabela-container">

<table class="tabela-premios">

<thead>

<tr>

<th>Concurso</th>

<th>Data</th>

<th>Local</th>

<th>Números Sorteados</th>

<th>Seu Jogo</th>

<th>Acertos</th>

<th>Prêmio</th>

<th>Status</th>

</tr>

</thead>

<tbody id="tabela-resultados">

<!-- Será preenchido via JavaScript -->

</tbody>

<tfoot id="tabela-totais">

<tr>

<td colspan="6"><strong>Total de Prêmios</strong></td>

<td colspan="2" class="total-premios"></td>

</tr>

</tfoot>

</table>

</div>

</div>

<div class="jogos-mais-sorteados">

<h2>Jogos Mais Sorteados</h2>

<div class="export-buttons hidden">

<button onclick="exportarDados('jogos-sorteados', 'xlsx')" class="btn-export">

<i class="fas fa-file-excel"></i> Excel

</button>

<button onclick="exportarDados('jogos-sorteados', 'html')" class="btn-export">

<i class="fas fa-file-code"></i> HTML

</button>

</div>

<div class="tabela-container">

<table class="tabela-premios" id="tabela-jogos-sorteados">

<thead>

<tr>

<th>Meu Jogo</th>

<th>Total de Acertos</th>

<th>Distribuição</th>

</tr>

</thead>

<tbody></tbody>

</table>

</div>

</div>

</div>

</div>

<script src="{{ url\_for('static', filename='js/main.js') }}"></script>

</body>

</html>

// // Variáveis globais

// const jogosIncluidos = [];

// const jogosSelecionados = new Set();

// let conferenciaCancelada = false;

// let dadosUltimaConsulta = null;

// // Configuração tamanho do lote

// const TAMANHO\_LOTE = 930;

// // Função para atualizar o contador e mensagem

// function atualizarContadorJogos() {

// const quantidade = jogosIncluidos.length;

// const contadorElement = document.getElementById('contador-jogos');

// if (contadorElement) {

// contadorElement.textContent = quantidade;

// }

// const tituloJogos = document.querySelector('.jogos-incluidos h3');

// if (tituloJogos) {

// tituloJogos.textContent = `Jogos Incluídos (${quantidade} ${quantidade === 1 ? 'jogo' : 'jogos'})`;

// }

// }

// // Função para formatar mensagens de jogos

// function formatarMensagemJogos(quantidade, acao) {

// if (acao === 'incluir') {

// return `${quantidade} jogo${quantidade === 1 ? ' foi incluído' : 's foram incluídos'} com sucesso!`;

// } else if (acao === 'remover') {

// return `${quantidade} jogo${quantidade === 1 ? ' foi removido' : 's foram removidos'} com sucesso!`;

// }

// return '';

// }

// async function combinarResultados(resultadosFinais, chunkResultados) {

// // Combina resumo

// resultadosFinais.resumo.quatro += chunkResultados.resumo.quatro;

// resultadosFinais.resumo.cinco += chunkResultados.resumo.cinco;

// resultadosFinais.resumo.seis += chunkResultados.resumo.seis;

// resultadosFinais.resumo.total\_premios += chunkResultados.resumo.total\_premios;

// // Combina acertos

// resultadosFinais.acertos.push(...chunkResultados.acertos);

// // Atualiza estatísticas dos jogos

// if (chunkResultados.jogos\_stats) {

// if (!resultadosFinais.jogos\_stats) {

// resultadosFinais.jogos\_stats = [];

// }

// chunkResultados.jogos\_stats.forEach(jogoStat => {

// const jogoExistente = resultadosFinais.jogos\_stats.find(

// js => JSON.stringify(js.numeros) === JSON.stringify(jogoStat.numeros)

// );

// if (jogoExistente) {

// jogoExistente.total += jogoStat.total;

// for (let i = 1; i <= 6; i++) {

// jogoExistente.distribuicao[i] += jogoStat.distribuicao[i];

// }

// } else {

// resultadosFinais.jogos\_stats.push(jogoStat);

// }

// });

// }

// }

// // Função para calcular os lotes

// function calcularLotes(inicio, fim) {

// const lotes = [];

// for (let i = inicio; i <= fim; i += TAMANHO\_LOTE) {

// const loteFim = Math.min(i + TAMANHO\_LOTE - 1, fim);

// lotes.push({inicio: i, fim: loteFim});

// }

// return lotes;

// }

// // Funções de Drag and Drop

// function setupDragAndDrop() {

// const dropZone = document.getElementById('drop-zone');

// const fileInput = document.getElementById('file-input');

// dropZone.onclick = () => fileInput.click();

// ['dragenter', 'dragover', 'dragleave', 'drop'].forEach(eventName => {

// dropZone.addEventListener(eventName, preventDefaults);

// });

// ['dragenter', 'dragover'].forEach(eventName => {

// dropZone.addEventListener(eventName, () => {

// dropZone.classList.add('dragover');

// });

// });

// ['dragleave', 'drop'].forEach(eventName => {

// dropZone.addEventListener(eventName, () => {

// dropZone.classList.remove('dragover');

// });

// });

// dropZone.addEventListener('drop', handleDrop);

// fileInput.addEventListener('change', handleFileSelect);

// // Adicione aqui o código para seleção dos números

// const numeros = document.querySelectorAll('.numero');

// numeros.forEach(numero => {

// numero.addEventListener('click', () => {

// const num = parseInt(numero.dataset.numero);

// if (numero.classList.contains('selecionado')) {

// numero.classList.remove('selecionado');

// numerosSelecionados.delete(num);

// } else if (numerosSelecionados.size < 6) {

// numero.classList.add('selecionado');

// numerosSelecionados.add(num);

// }

// });

// });

// }

// function preventDefaults(e) {

// e.preventDefault();

// e.stopPropagation();

// }

// async function handleDrop(e) {

// const file = e.dataTransfer.files[0];

// await processFile(file);

// }

// async function handleFileSelect(e) {

// const file = e.target.files[0];

// await processFile(file);

// }

// async function processFile(file) {

// if (!file) return;

// const dropZone = document.getElementById('drop-zone');

// dropZone.classList.add('processing');

// try {

// const formData = new FormData();

// formData.append('file', file);

// const response = await fetch('/processar\_arquivo', {

// method: 'POST',

// body: formData

// });

// const data = await response.json();

// if (data.error) {

// throw new Error(data.error);

// }

// if (data.jogos && data.jogos.length > 0) {

// const jogosAtuais = new Set(jogosIncluidos.map(j => JSON.stringify(j)));

// let jogosNovos = 0;

// data.jogos.forEach(jogo => {

// const jogoStr = JSON.stringify(jogo);

// if (!jogosAtuais.has(jogoStr)) {

// jogosIncluidos.push(jogo);

// adicionarJogoNaLista(jogo);

// jogosAtuais.add(jogoStr);

// jogosNovos++;

// }

// });

// atualizarContadorJogos();

// alert(formatarMensagemJogos(jogosNovos, 'incluir'));

// } else {

// alert('Nenhum jogo válido encontrado no arquivo');

// }

// } catch (error) {

// console.error('Erro:', error);

// alert(`Erro ao processar arquivo: ${error.message}`);

// } finally {

// dropZone.classList.remove('processing');

// }

// }

// // Funções de manipulação de jogos

// function adicionarJogoNaLista(jogo) {

// const jogoItem = document.createElement('div');

// jogoItem.className = 'jogo-item';

// const checkbox = document.createElement('input');

// checkbox.type = 'checkbox';

// checkbox.className = 'jogo-checkbox';

// checkbox.onclick = (e) => {

// const jogoStr = JSON.stringify(jogo);

// if (e.target.checked) {

// jogosSelecionados.add(jogoStr);

// jogoItem.classList.add('selecionado');

// } else {

// jogosSelecionados.delete(jogoStr);

// jogoItem.classList.remove('selecionado');

// }

// atualizarBotoesSeleção();

// };

// const jogoNumeros = document.createElement('div');

// jogoNumeros.className = 'jogo-numeros';

// jogo.forEach(num => {

// const numeroSpan = document.createElement('span');

// numeroSpan.className = 'jogo-numero';

// numeroSpan.textContent = String(num).padStart(2, '0');

// jogoNumeros.appendChild(numeroSpan);

// });

// const btnRemover = document.createElement('button');

// btnRemover.className = 'btn-remover';

// btnRemover.textContent = 'Remover';

// btnRemover.onclick = () => removerJogo(jogo, jogoItem);

// jogoItem.appendChild(checkbox);

// jogoItem.appendChild(jogoNumeros);

// jogoItem.appendChild(btnRemover);

// document.getElementById('lista-jogos').appendChild(jogoItem);

// }

// function removerJogo(jogo, jogoItem) {

// const index = jogosIncluidos.findIndex(j =>

// JSON.stringify(j) === JSON.stringify(jogo)

// );

// if (index !== -1) {

// jogosIncluidos.splice(index, 1);

// jogosSelecionados.delete(JSON.stringify(jogo));

// jogoItem.remove();

// atualizarBotoesSeleção();

// atualizarContadorJogos();

// alert(formatarMensagemJogos(1, 'remover'));

// }

// }

// function limparTodosJogos() {

// const quantidadeAtual = jogosIncluidos.length;

// if (quantidadeAtual === 0) {

// alert('Não há jogos para remover');

// return;

// }

// if (confirm('Tem certeza que deseja remover todos os jogos?')) {

// jogosIncluidos.length = 0;

// jogosSelecionados.clear();

// document.getElementById('lista-jogos').innerHTML = '';

// atualizarBotoesSeleção();

// atualizarContadorJogos();

// alert(formatarMensagemJogos(quantidadeAtual, 'remover'));

// }

// }

// function removerJogosSelecionados() {

// if (jogosSelecionados.size === 0) {

// alert('Selecione pelo menos um jogo para remover');

// return;

// }

// const quantidadeRemover = jogosSelecionados.size;

// if (confirm(`Deseja remover ${quantidadeRemover} jogo${quantidadeRemover === 1 ? '' : 's'} selecionado${quantidadeRemover === 1 ? '' : 's'}?`)) {

// jogosSelecionados.forEach(jogoStr => {

// const jogo = JSON.parse(jogoStr);

// const index = jogosIncluidos.findIndex(j =>

// JSON.stringify(j) === jogoStr

// );

// if (index !== -1) {

// jogosIncluidos.splice(index, 1);

// }

// });

// document.querySelectorAll('.jogo-checkbox:checked').forEach(checkbox => {

// checkbox.closest('.jogo-item').remove();

// });

// jogosSelecionados.clear();

// atualizarBotoesSeleção();

// atualizarContadorJogos();

// alert(formatarMensagemJogos(quantidadeRemover, 'remover'));

// }

// }

// function atualizarBotoesSeleção() {

// const removerSelecionadosBtn = document.getElementById('btn-remover-selecionados');

// if (removerSelecionadosBtn) {

// removerSelecionadosBtn.disabled = jogosSelecionados.size === 0;

// }

// }

// // Funções de exportação

// function toggleBotoesExportacao(mostrar) {

// document.querySelectorAll('.export-buttons').forEach(div => {

// if (mostrar) {

// div.classList.remove('hidden');

// } else {

// div.classList.add('hidden');

// }

// });

// }

// async function exportarDados(tipo, formato) {

// if (!dadosUltimaConsulta) {

// alert('Faça uma consulta primeiro antes de exportar os dados.');

// return;

// }

// try {

// const response = await fetch(`/exportar/${tipo}/${formato}`, {

// method: 'POST',

// headers: {

// 'Content-Type': 'application/json'

// },

// body: JSON.stringify(dadosUltimaConsulta)

// });

// if (!response.ok) {

// throw new Error('Erro ao exportar dados');

// }

// const blob = await response.blob();

// const url = window.URL.createObjectURL(blob);

// const a = document.createElement('a');

// a.href = url;

// a.download = `${tipo}.${formato}`;

// document.body.appendChild(a);

// a.click();

// document.body.removeChild(a);

// window.URL.revokeObjectURL(url);

// } catch (error) {

// console.error('Erro:', error);

// alert('Erro ao exportar os dados. Tente novamente.');

// }

// }

// // Funções de atualização de tabelas

// function atualizarTabelaJogosSorteados(jogos\_stats) {

// const tbody = document.querySelector('#tabela-jogos-sorteados tbody');

// tbody.innerHTML = '';

// jogos\_stats.forEach(jogo => {

// const tr = document.createElement('tr');

// const tdJogo = document.createElement('td');

// tdJogo.innerHTML = `<div class="numeros-tabela">

// ${jogo.numeros.map(n =>

// `<span class="numero-tabela">${String(n).padStart(2, '0')}</span>`

// ).join('')}

// </div>`;

// const tdTotal = document.createElement('td');

// tdTotal.textContent = `${jogo.total} vezes`;

// const tdDistribuicao = document.createElement('td');

// const distribuicao = [];

// for (let i = 1; i <= 6; i++) {

// if (jogo.distribuicao[i] > 0) {

// distribuicao.push(

// `<span class="distribuicao-badge">

// ${i} ponto${i !== 1 ? 's' : ''}: ${jogo.distribuicao[i]} vez${jogo.distribuicao[i] !== 1 ? 'es' : ''}

// </span>`

// );

// }

// }

// tdDistribuicao.innerHTML = distribuicao.join(' ');

// tr.appendChild(tdJogo);

// tr.appendChild(tdTotal);

// tr.appendChild(tdDistribuicao);

// tbody.appendChild(tr);

// });

// }

// function atualizarDetalhesETabela(data) {

// const detalhesDiv = document.getElementById('detalhes-resultados');

// const tabelaBody = document.getElementById('tabela-resultados');

// detalhesDiv.innerHTML = '';

// tabelaBody.innerHTML = '';

// data.acertos.forEach(resultado => {

// // Adicionar na seção de detalhes

// const resultadoDiv = document.createElement('div');

// resultadoDiv.className = 'resultado-item';

// resultadoDiv.innerHTML = `

// <div class="resultado-header">

// <h3>Concurso ${resultado.concurso} - ${resultado.data}</h3>

// <p>${resultado.local || ''}</p>

// </div>

// <div class="resultado-numeros">

// <div class="numeros-sorteados">

// <h4>Números Sorteados:</h4>

// <div class="numeros-lista">

// ${resultado.numeros\_sorteados

// .sort((a, b) => a - b)

// .map(n => `<span class="numero-sorteado">${String(n).padStart(2, '0')}</span>`)

// .join(' ')}

// </div>

// </div>

// <div class="seu-jogo">

// <h4>Seu Jogo:</h4>

// <div class="numeros-lista">

// ${resultado.seus\_numeros

// .sort((a, b) => a - b)

// .map(n => `<span class="numero-jogado ${resultado.numeros\_sorteados.includes(n) ? 'acerto' : ''}">${String(n).padStart(2, '0')}</span>`)

// .join(' ')}

// </div>

// </div>

// <div class="resultado-info">

// <p class="acertos-info">Acertos: <strong>${resultado.acertos}</strong></p>

// ${resultado.premio > 0 ?

// `<p class="premio-info">Prêmio: <strong>R$ ${resultado.premio.toLocaleString('pt-BR', { minimumFractionDigits: 2 })}</strong></p>`

// : ''}

// </div>

// </div>

// `;

// detalhesDiv.appendChild(resultadoDiv);

// // Adicionar na tabela

// const row = document.createElement('tr');

// const numerosSorteados = resultado.numeros\_sorteados

// .sort((a, b) => a - b)

// .map(n => `<span class="numero-tabela">${String(n).padStart(2, '0')}</span>`)

// .join('');

// const seusNumeros = resultado.seus\_numeros

// .sort((a, b) => a - b)

// .map(n => `<span class="numero-tabela ${resultado.numeros\_sorteados.includes(n) ? 'acerto' : ''}">${String(n).padStart(2, '0')}</span>`)

// .join('');

// const premioText = resultado.premio > 0

// ? `R$ ${resultado.premio.toLocaleString('pt-BR', {minimumFractionDigits: 2})}`

// : 'Não houve ganhadores';

// row.innerHTML = `

// <td>${resultado.concurso}</td>

// <td>${resultado.data}</td>

// <td>${resultado.local || '-'}</td>

// <td><div class="numeros-tabela">${numerosSorteados}</div></td>

// <td><div class="numeros-tabela">${seusNumeros}</div></td>

// <td>${resultado.acertos}</td>

// <td>${premioText}</td>

// <td>${resultado.premio > 0 ? 'Premiado' : 'Acumulado'}</td>

// `;

// tabelaBody.appendChild(row);

// });

// // Atualiza o total no rodapé

// const totalCell = document.querySelector('.total-premios');

// if (totalCell && data.resumo.total\_premios) {

// totalCell.textContent = `R$ ${data.resumo.total\_premios.toLocaleString('pt-BR', {minimumFractionDigits: 2})}`;

// }

// }

// // Inicialização do documento

// document.addEventListener('DOMContentLoaded', function() {

// setupDragAndDrop();

// const numeros = document.querySelectorAll('.numero');

// const limparBtn = document.getElementById('limpar');

// const sugestaoBtn = document.getElementById('sugestao');

// const conferirBtn = document.getElementById('conferir');

// const incluirBtn = document.getElementById('incluir');

// const listaJogos = document.getElementById('lista-jogos');

// const overlay = document.getElementById('overlay');

// const numerosSelecionados = new Set();

// // Ocultar botões de exportação inicialmente

// toggleBotoesExportacao(false);

// // Configuração do botão de cancelar conferência

// const btnCancelarConferencia = document.getElementById('btn-cancelar-conferencia');

// if (btnCancelarConferencia) {

// btnCancelarConferencia.addEventListener('click', () => {

// conferenciaCancelada = true;

// overlay.style.display = 'none';

// });

// }

// // Configuração dos botões de ação

// document.getElementById('btn-limpar-todos').addEventListener('click', limparTodosJogos);

// document.getElementById('btn-remover-selecionados').addEventListener('click', removerJogosSelecionados);

// // Inicialização

// atualizarContadorJogos();

// // Eventos de seleção de números

// numeros.forEach(numero => {

// numero.addEventListener('click', () => {

// const num = parseInt(numero.dataset.numero);

// if (numero.classList.contains('selecionado')) {

// numero.classList.remove('selecionado');

// numerosSelecionados.delete(num);

// } else if (numerosSelecionados.size < 6) {

// numero.classList.add('selecionado');

// numerosSelecionados.add(num);

// }

// });

// });

// // Botão Limpar

// limparBtn.addEventListener('click', () => {

// numeros.forEach(numero => numero.classList.remove('selecionado'));

// numerosSelecionados.clear();

// });

// // Botão Sugestão

// sugestaoBtn.addEventListener('click', async () => {

// const response = await fetch('/gerar\_numeros');

// const data = await response.json();

// limparBtn.click();

// data.numeros.forEach(num => {

// const numero = document.querySelector(`[data-numero="${num}"]`);

// numero.classList.add('selecionado');

// numerosSelecionados.add(num);

// });

// });

// // Botão Incluir

// incluirBtn.addEventListener('click', () => {

// if (numerosSelecionados.size !== 6) {

// alert('Selecione 6 números antes de incluir o jogo!');

// return;

// }

// const numerosArray = Array.from(numerosSelecionados).sort((a, b) => a - b);

// jogosIncluidos.push(numerosArray);

// adicionarJogoNaLista(numerosArray);

// atualizarContadorJogos();

// alert('1 jogo foi incluído com sucesso!');

// limparBtn.click();

// });

// // Botão Conferir

// conferirBtn.addEventListener('click', async () => {

// if (jogosIncluidos.length === 0) {

// alert('Inclua pelo menos um jogo antes de conferir!');

// return;

// }

// const inicio = parseInt(document.getElementById('inicio').value);

// const fim = parseInt(document.getElementById('fim').value);

// if (!inicio || !fim || inicio > fim) {

// alert('Verifique os números dos concursos!');

// return;

// }

// overlay.style.display = 'flex';

// conferenciaCancelada = false;

// toggleBotoesExportacao(false);

// try {

// const resultadosFinais = {

// acertos: [],

// resumo: { quatro: 0, cinco: 0, seis: 0, total\_premios: 0 },

// jogos\_stats: []

// };

// // Processa jogos em chunks

// const CHUNK\_SIZE = 5000;

// for (let i = 0; i < jogosIncluidos.length && !conferenciaCancelada; i += CHUNK\_SIZE) {

// const chunkJogos = jogosIncluidos.slice(i, i + CHUNK\_SIZE);

// const progresso = ((i / jogosIncluidos.length) \* 100).toFixed(1);

// document.querySelector('.progress-text').textContent =

// `Processando ${progresso}% dos jogos (${i + 1} até ${Math.min(i + CHUNK\_SIZE, jogosIncluidos.length)})...`;

// // Processa cada chunk em lotes de concursos

// const lotes = calcularLotes(inicio, fim);

// for (const lote of lotes) {

// if (conferenciaCancelada) break;

// document.querySelector('.progress-text').textContent =

// `Processando concursos ${lote.inicio} a ${lote.fim} (${progresso}% dos jogos)...`;

// const response = await fetch('/conferir', {

// method: 'POST',

// headers: { 'Content-Type': 'application/json' },

// body: JSON.stringify({

// inicio: lote.inicio,

// fim: lote.fim,

// jogos: chunkJogos

// })

// });

// if (!response.ok) throw new Error('Erro ao processar jogos');

// const chunkResultados = await response.json();

// await combinarResultados(resultadosFinais, chunkResultados);

// await atualizarInterfaceProgressiva(resultadosFinais);

// }

// }

// dadosUltimaConsulta = resultadosFinais;

// toggleBotoesExportacao(true);

// } catch (error) {

// console.error('Erro:', error);

// alert('Ocorreu um erro ao processar os jogos. Tente novamente.');

// } finally {

// overlay.style.display = 'none';

// }

// });

// async function atualizarInterfaceProgressiva(resultados) {

// document.getElementById('quatro-acertos').textContent = resultados.resumo.quatro;

// document.getElementById('cinco-acertos').textContent = resultados.resumo.cinco;

// document.getElementById('seis-acertos').textContent = resultados.resumo.seis;

// const total\_premios = resultados.resumo.total\_premios;

// const premiosCard = {

// quatro: resultados.acertos.filter(r => r.acertos === 4).reduce((sum, r) => sum + r.premio, 0),

// cinco: resultados.acertos.filter(r => r.acertos === 5).reduce((sum, r) => sum + r.premio, 0),

// seis: resultados.acertos.filter(r => r.acertos === 6).reduce((sum, r) => sum + r.premio, 0)

// };

// document.getElementById('quatro-valor').textContent = premiosCard.quatro > 0 ?

// `R$ ${premiosCard.quatro.toLocaleString('pt-BR', {minimumFractionDigits: 2})}` :

// 'Não houve ganhadores';

// document.getElementById('cinco-valor').textContent = premiosCard.cinco > 0 ?

// `R$ ${premiosCard.cinco.toLocaleString('pt-BR', {minimumFractionDigits: 2})}` :

// 'Não houve ganhadores';

// document.getElementById('seis-valor').textContent = premiosCard.seis > 0 ?

// `R$ ${premiosCard.seis.toLocaleString('pt-BR', {minimumFractionDigits: 2})}` :

// 'Não houve ganhadores';

// if (resultados.acertos && resultados.acertos.length > 0) {

// atualizarDetalhesETabela(resultados);

// toggleBotoesExportacao(true);

// }

// if (resultados.jogos\_stats) {

// atualizarTabelaJogosSorteados(resultados.jogos\_stats);

// }

// }

// //ALTERADO 24/01/2025

// // Função para calcular lotes

// function calcularLotes(inicio, fim) {

// const lotes = [];

// for (let i = inicio; i <= fim; i += TAMANHO\_LOTE) {

// const loteFim = Math.min(i + TAMANHO\_LOTE - 1, fim);

// lotes.push({inicio: i, fim: loteFim});

// }

// return lotes;

// };

// Variáveis globais

const jogosIncluidos = [];

const jogosSelecionados = new Set();

const numerosSelecionados = new Set();

let conferenciaCancelada = false;

let dadosUltimaConsulta = null;

// Configuração tamanho do lote

const TAMANHO\_LOTE = 930;

// Função para atualizar o contador e mensagem

function atualizarContadorJogos() {

const quantidade = jogosIncluidos.length;

const contadorElement = document.getElementById('contador-jogos');

if (contadorElement) {

contadorElement.textContent = quantidade;

}

const tituloJogos = document.querySelector('.jogos-incluidos h3');

if (tituloJogos) {

tituloJogos.textContent = `Jogos Incluídos (${quantidade} ${quantidade === 1 ? 'jogo' : 'jogos'})`;

}

}

// Função para formatar mensagens de jogos

function formatarMensagemJogos(quantidade, acao) {

if (acao === 'incluir') {

return `${quantidade} jogo${quantidade === 1 ? ' foi incluído' : 's foram incluídos'} com sucesso!`;

} else if (acao === 'remover') {

return `${quantidade} jogo${quantidade === 1 ? ' foi removido' : 's foram removidos'} com sucesso!`;

}

return '';

}

async function combinarResultados(resultadosFinais, chunkResultados) {

// Combina resumo

resultadosFinais.resumo.quatro += chunkResultados.resumo.quatro;

resultadosFinais.resumo.cinco += chunkResultados.resumo.cinco;

resultadosFinais.resumo.seis += chunkResultados.resumo.seis;

resultadosFinais.resumo.total\_premios += chunkResultados.resumo.total\_premios;

// Combina acertos

resultadosFinais.acertos.push(...chunkResultados.acertos);

// Atualiza estatísticas dos jogos

if (chunkResultados.jogos\_stats) {

if (!resultadosFinais.jogos\_stats) {

resultadosFinais.jogos\_stats = [];

}

chunkResultados.jogos\_stats.forEach(jogoStat => {

const jogoExistente = resultadosFinais.jogos\_stats.find(

js => JSON.stringify(js.numeros) === JSON.stringify(jogoStat.numeros)

);

if (jogoExistente) {

jogoExistente.total += jogoStat.total;

for (let i = 1; i <= 6; i++) {

jogoExistente.distribuicao[i] += jogoStat.distribuicao[i];

}

} else {

resultadosFinais.jogos\_stats.push(jogoStat);

}

});

}

}

// Função para calcular os lotes

function calcularLotes(inicio, fim) {

const lotes = [];

for (let i = inicio; i <= fim; i += TAMANHO\_LOTE) {

const loteFim = Math.min(i + TAMANHO\_LOTE - 1, fim);

lotes.push({inicio: i, fim: loteFim});

}

return lotes;

}

// Funções de Drag and Drop

function setupDragAndDrop() {

const dropZone = document.getElementById('drop-zone');

const fileInput = document.getElementById('file-input');

dropZone.onclick = () => fileInput.click();

['dragenter', 'dragover', 'dragleave', 'drop'].forEach(eventName => {

dropZone.addEventListener(eventName, preventDefaults);

});

['dragenter', 'dragover'].forEach(eventName => {

dropZone.addEventListener(eventName, () => {

dropZone.classList.add('dragover');

});

});

['dragleave', 'drop'].forEach(eventName => {

dropZone.addEventListener(eventName, () => {

dropZone.classList.remove('dragover');

});

});

dropZone.addEventListener('drop', handleDrop);

fileInput.addEventListener('change', handleFileSelect);

}

function preventDefaults(e) {

e.preventDefault();

e.stopPropagation();

}

async function handleDrop(e) {

const file = e.dataTransfer.files[0];

await processFile(file);

}

async function handleFileSelect(e) {

const file = e.target.files[0];

await processFile(file);

}

async function processFile(file) {

if (!file) return;

const dropZone = document.getElementById('drop-zone');

dropZone.classList.add('processing');

try {

const formData = new FormData();

formData.append('file', file);

const response = await fetch('/processar\_arquivo', {

method: 'POST',

body: formData

});

const data = await response.json();

if (data.error) {

throw new Error(data.error);

}

if (data.jogos && data.jogos.length > 0) {

const jogosAtuais = new Set(jogosIncluidos.map(j => JSON.stringify(j)));

let jogosNovos = 0;

data.jogos.forEach(jogo => {

const jogoStr = JSON.stringify(jogo);

if (!jogosAtuais.has(jogoStr)) {

jogosIncluidos.push(jogo);

adicionarJogoNaLista(jogo);

jogosAtuais.add(jogoStr);

jogosNovos++;

}

});

atualizarContadorJogos();

alert(formatarMensagemJogos(jogosNovos, 'incluir'));

} else {

alert('Nenhum jogo válido encontrado no arquivo');

}

} catch (error) {

console.error('Erro:', error);

alert(`Erro ao processar arquivo: ${error.message}`);

} finally {

dropZone.classList.remove('processing');

}

}

// Funções de manipulação de jogos

function adicionarJogoNaLista(jogo) {

const jogoItem = document.createElement('div');

jogoItem.className = 'jogo-item';

const checkbox = document.createElement('input');

checkbox.type = 'checkbox';

checkbox.className = 'jogo-checkbox';

checkbox.onclick = (e) => {

const jogoStr = JSON.stringify(jogo);

if (e.target.checked) {

jogosSelecionados.add(jogoStr);

jogoItem.classList.add('selecionado');

} else {

jogosSelecionados.delete(jogoStr);

jogoItem.classList.remove('selecionado');

}

atualizarBotoesSeleção();

};

const jogoNumeros = document.createElement('div');

jogoNumeros.className = 'jogo-numeros';

jogo.forEach(num => {

const numeroSpan = document.createElement('span');

numeroSpan.className = 'jogo-numero';

numeroSpan.textContent = String(num).padStart(2, '0');

jogoNumeros.appendChild(numeroSpan);

});

const btnRemover = document.createElement('button');

btnRemover.className = 'btn-remover';

btnRemover.textContent = 'Remover';

btnRemover.onclick = () => removerJogo(jogo, jogoItem);

jogoItem.appendChild(checkbox);

jogoItem.appendChild(jogoNumeros);

jogoItem.appendChild(btnRemover);

document.getElementById('lista-jogos').appendChild(jogoItem);

}

function removerJogo(jogo, jogoItem) {

const index = jogosIncluidos.findIndex(j =>

JSON.stringify(j) === JSON.stringify(jogo)

);

if (index !== -1) {

jogosIncluidos.splice(index, 1);

jogosSelecionados.delete(JSON.stringify(jogo));

jogoItem.remove();

atualizarBotoesSeleção();

atualizarContadorJogos();

alert(formatarMensagemJogos(1, 'remover'));

}

}

function limparTodosJogos() {

const quantidadeAtual = jogosIncluidos.length;

if (quantidadeAtual === 0) {

alert('Não há jogos para remover');

return;

}

if (confirm('Tem certeza que deseja remover todos os jogos?')) {

jogosIncluidos.length = 0;

jogosSelecionados.clear();

document.getElementById('lista-jogos').innerHTML = '';

atualizarBotoesSeleção();

atualizarContadorJogos();

alert(formatarMensagemJogos(quantidadeAtual, 'remover'));

}

}

function removerJogosSelecionados() {

if (jogosSelecionados.size === 0) {

alert('Selecione pelo menos um jogo para remover');

return;

}

const quantidadeRemover = jogosSelecionados.size;

if (confirm(`Deseja remover ${quantidadeRemover} jogo${quantidadeRemover === 1 ? '' : 's'} selecionado${quantidadeRemover === 1 ? '' : 's'}?`)) {

jogosSelecionados.forEach(jogoStr => {

const jogo = JSON.parse(jogoStr);

const index = jogosIncluidos.findIndex(j =>

JSON.stringify(j) === jogoStr

);

if (index !== -1) {

jogosIncluidos.splice(index, 1);

}

});

document.querySelectorAll('.jogo-checkbox:checked').forEach(checkbox => {

checkbox.closest('.jogo-item').remove();

});

jogosSelecionados.clear();

atualizarBotoesSeleção();

atualizarContadorJogos();

alert(formatarMensagemJogos(quantidadeRemover, 'remover'));

}

}

function atualizarBotoesSeleção() {

const removerSelecionadosBtn = document.getElementById('btn-remover-selecionados');

if (removerSelecionadosBtn) {

removerSelecionadosBtn.disabled = jogosSelecionados.size === 0;

}

}

// Funções de exportação

function toggleBotoesExportacao(mostrar) {

document.querySelectorAll('.export-buttons').forEach(div => {

if (mostrar) {

div.classList.remove('hidden');

} else {

div.classList.add('hidden');

}

});

}

async function exportarDados(tipo, formato) {

if (!dadosUltimaConsulta) {

alert('Faça uma consulta primeiro antes de exportar os dados.');

return;

}

try {

const response = await fetch(`/exportar/${tipo}/${formato}`, {

method: 'POST',

headers: {

'Content-Type': 'application/json'

},

body: JSON.stringify(dadosUltimaConsulta)

});

if (!response.ok) {

throw new Error('Erro ao exportar dados');

}

const blob = await response.blob();

const url = window.URL.createObjectURL(blob);

const a = document.createElement('a');

a.href = url;

a.download = `${tipo}.${formato}`;

document.body.appendChild(a);

a.click();

document.body.removeChild(a);

window.URL.revokeObjectURL(url);

} catch (error) {

console.error('Erro:', error);

alert('Erro ao exportar os dados. Tente novamente.');

}

}

// Funções de atualização de tabelas

function atualizarTabelaJogosSorteados(jogos\_stats) {

const tbody = document.querySelector('#tabela-jogos-sorteados tbody');

tbody.innerHTML = '';

jogos\_stats.forEach(jogo => {

const tr = document.createElement('tr');

const tdJogo = document.createElement('td');

tdJogo.innerHTML = `<div class="numeros-tabela">

${jogo.numeros.map(n =>

`<span class="numero-tabela">${String(n).padStart(2, '0')}</span>`

).join('')}

</div>`;

const tdTotal = document.createElement('td');

tdTotal.textContent = `${jogo.total} vezes`;

const tdDistribuicao = document.createElement('td');

const distribuicao = [];

for (let i = 1; i <= 6; i++) {

if (jogo.distribuicao[i] > 0) {

distribuicao.push(

`<span class="distribuicao-badge">

${i} ponto${i !== 1 ? 's' : ''}: ${jogo.distribuicao[i]} vez${jogo.distribuicao[i] !== 1 ? 'es' : ''}

</span>`

);

}

}

tdDistribuicao.innerHTML = distribuicao.join(' ');

tr.appendChild(tdJogo);

tr.appendChild(tdTotal);

tr.appendChild(tdDistribuicao);

tbody.appendChild(tr);

});

}

function atualizarDetalhesETabela(data) {

const detalhesDiv = document.getElementById('detalhes-resultados');

const tabelaBody = document.getElementById('tabela-resultados');

detalhesDiv.innerHTML = '';

tabelaBody.innerHTML = '';

data.acertos.forEach(resultado => {

// Adicionar na seção de detalhes

const resultadoDiv = document.createElement('div');

resultadoDiv.className = 'resultado-item';

resultadoDiv.innerHTML = `

<div class="resultado-header">

<h3>Concurso ${resultado.concurso} - ${resultado.data}</h3>

<p>${resultado.local || ''}</p>

</div>

<div class="resultado-numeros">

<div class="numeros-sorteados">

<h4>Números Sorteados:</h4>

<div class="numeros-lista">

${resultado.numeros\_sorteados

.sort((a, b) => a - b)

.map(n => `<span class="numero-sorteado">${String(n).padStart(2, '0')}</span>`)

.join(' ')}

</div>

</div>

<div class="seu-jogo">

<h4>Seu Jogo:</h4>

<div class="numeros-lista">

${resultado.seus\_numeros

.sort((a, b) => a - b)

.map(n => `<span class="numero-jogado ${resultado.numeros\_sorteados.includes(n) ? 'acerto' : ''}">${String(n).padStart(2, '0')}</span>`)

.join(' ')}

</div>

</div>

<div class="resultado-info">

<p class="acertos-info">Acertos: <strong>${resultado.acertos}</strong></p>

${resultado.premio > 0 ?

`<p class="premio-info">Prêmio: <strong>R$ ${resultado.premio.toLocaleString('pt-BR', { minimumFractionDigits: 2 })}</strong></p>`

: ''}

</div>

</div>

`;

detalhesDiv.appendChild(resultadoDiv);

// Adicionar na tabela

const row = document.createElement('tr');

const numerosSorteados = resultado.numeros\_sorteados

.sort((a, b) => a - b)

.map(n => `<span class="numero-tabela">${String(n).padStart(2, '0')}</span>`)

.join('');

const seusNumeros = resultado.seus\_numeros

.sort((a, b) => a - b)

.map(n => `<span class="numero-tabela ${resultado.numeros\_sorteados.includes(n) ? 'acerto' : ''}">${String(n).padStart(2, '0')}</span>`)

.join('');

const premioText = resultado.premio > 0

? `R$ ${resultado.premio.toLocaleString('pt-BR', {minimumFractionDigits: 2})}`

: 'Não houve ganhadores';

row.innerHTML = `

<td>${resultado.concurso}</td>

<td>${resultado.data}</td>

<td>${resultado.local || '-'}</td>

<td><div class="numeros-tabela">${numerosSorteados}</div></td>

<td><div class="numeros-tabela">${seusNumeros}</div></td>

<td>${resultado.acertos}</td>

<td>${premioText}</td>

<td>${resultado.premio > 0 ? 'Premiado' : 'Acumulado'}</td>

`;

tabelaBody.appendChild(row);

});

// Atualiza o total no rodapé

const totalCell = document.querySelector('.total-premios');

if (totalCell && data.resumo.total\_premios) {

totalCell.textContent = `R$ ${data.resumo.total\_premios.toLocaleString('pt-BR', {minimumFractionDigits: 2})}`;

}

}

// Inicialização do documento

document.addEventListener('DOMContentLoaded', function() {

setupDragAndDrop();

const numeros = document.querySelectorAll('.numero');

const limparBtn = document.getElementById('limpar');

const sugestaoBtn = document.getElementById('sugestao');

const conferirBtn = document.getElementById('conferir');

const incluirBtn = document.getElementById('incluir');

const listaJogos = document.getElementById('lista-jogos');

const overlay = document.getElementById('overlay');

// Ocultar botões de exportação inicialmente

toggleBotoesExportacao(false);

// Configuração do botão de cancelar conferência

const btnCancelarConferencia = document.getElementById('btn-cancelar-conferencia');

if (btnCancelarConferencia) {

btnCancelarConferencia.addEventListener('click', () => {

conferenciaCancelada = true;

overlay.style.display = 'none';

});

}

// Configuração dos botões de ação

document.getElementById('btn-limpar-todos').addEventListener('click', limparTodosJogos);

document.getElementById('btn-remover-selecionados').addEventListener('click', removerJogosSelecionados);

// Inicialização

atualizarContadorJogos();

// Eventos de seleção de números

numeros.forEach(numero => {

numero.addEventListener('click', () => {

const num = parseInt(numero.dataset.numero);

if (numero.classList.contains('selecionado')) {

numero.classList.remove('selecionado');

numerosSelecionados.delete(num);

} else if (numerosSelecionados.size < 6) {

numero.classList.add('selecionado');

numerosSelecionados.add(num);

}

});

});

// Botão Limpar

limparBtn.addEventListener('click', () => {

numeros.forEach(numero => numero.classList.remove('selecionado'));

numerosSelecionados.clear();

});

// Botão Sugestão

sugestaoBtn.addEventListener('click', async () => {

const response = await fetch('/gerar\_numeros');

const data = await response.json();

limparBtn.click();

data.numeros.forEach(num => {

const numero = document.querySelector(`[data-numero="${num}"]`);

numero.classList.add('selecionado');

numerosSelecionados.add(num);

});

});

// Botão Incluir

incluirBtn.addEventListener('click', () => {

if (numerosSelecionados.size !== 6) {

alert('Selecione 6 números antes de incluir o jogo!');

return;

}

const numerosArray = Array.from(numerosSelecionados).sort((a, b) => a - b);

jogosIncluidos.push(numerosArray);

adicionarJogoNaLista(numerosArray);

atualizarContadorJogos();

alert('1 jogo foi incluído com sucesso!');

limparBtn.click();

});

// // Botão Conferir

// conferirBtn.addEventListener('click', async () => {

// if (jogosIncluidos.length === 0) {

// alert('Inclua pelo menos um jogo antes de conferir!');

// return;

// }

// const inicio = parseInt(document.getElementById('inicio').value);

// const fim = parseInt(document.getElementById('fim').value);

// if (!inicio || !fim || inicio > fim) {

// alert('Verifique os números dos concursos!');

// return;

// }

// overlay.style.display = 'flex';

// conferenciaCancelada = false;

// toggleBotoesExportacao(false);

// try {

// const resultadosFinais = {

// acertos: [],

// resumo: { quatro: 0, cinco: 0, seis: 0, total\_premios: 0 },

// jogos\_stats: []

// };

// // Processa jogos em chunks

// const CHUNK\_SIZE = 5000;

// for (let i = 0; i < jogosIncluidos.length && !conferenciaCancelada; i += CHUNK\_SIZE) {

// const chunkJogos = jogosIncluidos.slice(i, i + CHUNK\_SIZE);

// const progresso = ((i / jogosIncluidos.length) \* 100).toFixed(1);

// document.querySelector('.progress-text').textContent =

// `Processando ${progresso}% dos jogos (${i + 1} até ${Math.min(i + CHUNK\_SIZE, jogosIncluidos.length)})...`;

// // Processa cada chunk em lotes de concursos

// const lotes = calcularLotes(inicio, fim);

// for (const lote of lotes) {

// if (conferenciaCancelada) break;

// document.querySelector('.progress-text').textContent =

// `Processando concursos ${lote.inicio} a ${lote.fim} (${progresso}% dos jogos)...`;

// const response = await fetch('/conferir', {

// method: 'POST',

// headers: { 'Content-Type': 'application/json' },

// body: JSON.stringify({

// inicio: lote.inicio,

// fim: lote.fim,

// jogos: chunkJogos

// })

// });

// if (!response.ok) throw new Error('Erro ao processar jogos');

// const chunkResultados = await response.json();

// await combinarResultados(resultadosFinais, chunkResultados);

// await atualizarInterfaceProgressiva(resultadosFinais);

// }

// }

// dadosUltimaConsulta = resultadosFinais;

// toggleBotoesExportacao(true);

// } catch (error) {

// console.error('Erro:', error);

// alert('Ocorreu um erro ao processar os jogos. Tente novamente.');

// } finally {

// overlay.style.display = 'none';

// }

// });

// });

// Botão Conferir

conferirBtn.addEventListener('click', async () => {

if (jogosIncluidos.length === 0) {

alert('Inclua pelo menos um jogo antes de conferir!');

return;

}

const inicio = parseInt(document.getElementById('inicio').value);

const fim = parseInt(document.getElementById('fim').value);

if (!inicio || !fim || inicio > fim) {

alert('Verifique os números dos concursos!');

return;

}

overlay.style.display = 'flex';

conferenciaCancelada = false;

toggleBotoesExportacao(false);

let response;

try {

const resultadosFinais = {

acertos: [],

resumo: { quatro: 0, cinco: 0, seis: 0, total\_premios: 0 },

jogos\_stats: []

};

// Processa jogos em chunks

const CHUNK\_SIZE = 5000;

for (let i = 0; i < jogosIncluidos.length && !conferenciaCancelada; i += CHUNK\_SIZE) {

const chunkJogos = jogosIncluidos.slice(i, i + CHUNK\_SIZE);

const progresso = ((i / jogosIncluidos.length) \* 100).toFixed(1);

document.querySelector('.progress-text').textContent =

`Processando ${progresso}% dos jogos (${i + 1} até ${Math.min(i + CHUNK\_SIZE, jogosIncluidos.length)})...`;

// Processa cada chunk em lotes de concursos

const lotes = calcularLotes(inicio, fim);

for (const lote of lotes) {

if (conferenciaCancelada) break;

document.querySelector('.progress-text').textContent =

`Processando concursos ${lote.inicio} a ${lote.fim} (${progresso}% dos jogos)...`;

const response = await fetch('/conferir', {

method: 'POST',

headers: { 'Content-Type': 'application/json' },

body: JSON.stringify({

inicio: lote.inicio,

fim: lote.fim,

jogos: chunkJogos

})

});

if (!response.ok) throw new Error('Erro ao processar jogos');

const chunkResultados = await response.json();

await combinarResultados(resultadosFinais, chunkResultados);

await atualizarInterfaceProgressiva(resultadosFinais);

}

}

dadosUltimaConsulta = resultadosFinais;

toggleBotoesExportacao(true);

} catch (error) {

console.error('Erro detalhado:', {

mensagem: error.message,

pilha: error.stack,

status: response?.status,

resposta: await response?.text()

});

let mensagemErro = 'Ocorreu um erro ao processar os jogos.';

if (error.message.includes('Failed to fetch')) {

mensagemErro += ' Verifique sua conexão com o servidor.';

} else if (response?.status === 413) {

mensagemErro += ' Quantidade de dados excede o limite.';

} else if (response?.status === 429) {

mensagemErro += ' Muitas requisições, aguarde um momento.';

}

alert(mensagemErro);

} finally {

overlay.style.display = 'none';

conferenciaCancelada = false;

}

});

});

async function atualizarInterfaceProgressiva(resultados) {

document.getElementById('quatro-acertos').textContent = resultados.resumo.quatro;

document.getElementById('cinco-acertos').textContent = resultados.resumo.cinco;

document.getElementById('seis-acertos').textContent = resultados.resumo.seis;

const total\_premios = resultados.resumo.total\_premios;

const premiosCard = {

quatro: resultados.acertos.filter(r => r.acertos === 4).reduce((sum, r) => sum + r.premio, 0),

cinco: resultados.acertos.filter(r => r.acertos === 5).reduce((sum, r) => sum + r.premio, 0),

seis: resultados.acertos.filter(r => r.acertos === 6).reduce((sum, r) => sum + r.premio, 0)

};

document.getElementById('quatro-valor').textContent = premiosCard.quatro > 0 ?

`R$ ${premiosCard.quatro.toLocaleString('pt-BR', {minimumFractionDigits: 2})}` :

'Não houve ganhadores';

document.getElementById('cinco-valor').textContent = premiosCard.cinco > 0 ?

`R$ ${premiosCard.cinco.toLocaleString('pt-BR', {minimumFractionDigits: 2})}` :

'Não houve ganhadores';

document.getElementById('seis-valor').textContent = premiosCard.seis > 0 ?

`R$ ${premiosCard.seis.toLocaleString('pt-BR', {minimumFractionDigits: 2})}` :

'Não houve ganhadores';

if (resultados.acertos && resultados.acertos.length > 0) {

atualizarDetalhesETabela(resultados);

toggleBotoesExportacao(true);

}

if (resultados.jogos\_stats) {

atualizarTabelaJogosSorteados(resultados.jogos\_stats);

}

}

// Continuação...

// ALTERADO 24/01/2025

// Função para calcular lotes

function calcularLotes(inicio, fim) {

const lotes = [];

for (let i = inicio; i <= fim; i += TAMANHO\_LOTE) {

const loteFim = Math.min(i + TAMANHO\_LOTE - 1, fim);

lotes.push({inicio: i, fim: loteFim});

}

return lotes;

}

:root {

--primary-color: #008751;

--secondary-color: #005c36;

--danger-color: #dc3545;

--success-color: #28a745;

}

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

}

body {

font-family: Arial, sans-serif;

line-height: 1.6;

padding: 20px;

background-color: #f4f4f4;

}

.container {

max-width: 1200px;

margin: 0 auto;

background-color: white;

padding: 20px;

border-radius: 8px;

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

/\*text-align: center;\*/

}

.volante {

margin: 20px 0;

padding: 20px;

background-color: #f8f9fa;

border-radius: 8px;

text-align: center;

}

.volante h2 {

color: #495057;

margin-bottom: 20px;

font-size: 1.2rem;

}

.numeros {

display: grid;

grid-template-columns: repeat(10, 1fr);

gap: 8px;

max-width: 800px;

margin: 0 auto;

padding: 15px;

}

.numero {

width: 40px;

height: 40px;

border-radius: 50%;

border: 2px solid #ced4da;

background-color: white;

color: #495057;

font-weight: bold;

font-size: 16px;

cursor: pointer;

transition: all 0.2s ease;

display: flex;

align-items: center;

justify-content: center;

padding: 0;

margin: 0 auto;

}

.numero:hover {

background-color: #e9ecef;

border-color: #adb5bd;

}

.numero.selecionado {

background-color: #28a745;

color: white;

border-color: #28a745;

}

.drop-zone {

width: 100%;

max-width: 800px;

height: 180px;

margin: 20px auto;

padding: 25px;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

text-align: center;

font-size: 1.2em;

color: #008751;

border: 3px dashed #008751;

border-radius: 12px;

background-color: rgba(0, 135, 81, 0.05);

transition: all 0.3s ease;

cursor: pointer;

position: relative;

overflow: hidden;

}

.drop-zone:hover {

transform: translateY(-2px);

box-shadow: 0 4px 12px rgba(0, 135, 81, 0.2);

background-color: rgba(0, 135, 81, 0.1);

}

.drop-zone.dragover {

background-color: rgba(40, 167, 69, 0.15);

border-color: #28a745;

transform: scale(1.02);

box-shadow: 0 6px 16px rgba(40, 167, 69, 0.25);

}

.drop-zone.dragover .drop-zone-content {

transform: scale(1.05);

}

.drop-zone-content {

transition: transform 0.3s ease;

}

.drop-zone-content p {

margin: 0;

color: #008751;

font-weight: 500;

}

.drop-zone-hint {

font-size: 0.9em;

color: #666;

margin-top: 10px;

opacity: 0.8;

}

.drop-zone.processing {

animation: pulse 2s infinite;

pointer-events: none;

background-color: rgba(0, 135, 81, 0.1);

}

.drop-zone.processing .drop-zone-content {

opacity: 0.5;

}

.drop-zone.processing::after {

content: "Processando...";

position: absolute;

top: 50%;

left: 50%;

transform: translate(-50%, -50%);

font-weight: bold;

color: #008751;

z-index: 2;

}

.drop-zone::before {

content: "⬆️";

font-size: 2em;

margin-bottom: 10px;

display: block;

transition: transform 0.3s ease;

}

.drop-zone.dragover::before {

transform: scale(1.2) translateY(-5px);

}

.controles {

display: flex;

gap: 20px;

margin: 20px 0;

padding: 20px;

background-color: #f8f9fa;

border-radius: 8px;

align-items: center;

}

.input-group {

display: flex;

flex-direction: column;

gap: 5px;

}

.input-group label {

font-weight: bold;

color: #495057;

}

.input-group input {

padding: 8px;

border: 1px solid #ced4da;

border-radius: 4px;

width: 120px;

}

.checkbox-group {

display: flex;

align-items: center;

gap: 5px;

}

.acoes {

display: flex;

gap: 10px;

margin: 20px 0;

justify-content: center;

}

.btn {

padding: 10px 20px;

border: none;

border-radius: 4px;

cursor: pointer;

font-weight: bold;

transition: all 0.3s ease;

}

.btn:hover {

opacity: 0.9;

transform: translateY(-1px);

}

.btn-vermelho {

background-color: var(--danger-color);

color: white;

}

.btn-azul {

background-color: #007bff;

color: white;

}

.btn-roxo {

background-color: #6f42c1;

color: white;

}

.btn-verde {

background-color: var(--success-color);

color: white;

}

.cards {

display: grid;

grid-template-columns: repeat(3, 1fr);

gap: 20px;

margin-top: 20px;

}

.card {

padding: 20px;

border-radius: 8px;

background-color: #f8f9fa;

text-align: center;

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.05);

}

.card h3 {

color: #495057;

margin-bottom: 10px;

}

.contagem {

font-size: 24px;

font-weight: bold;

color: var(--primary-color);

}

.lista-jogos {

margin-top: 20px;

border: 1px solid #ddd;

border-radius: 4px;

padding: 10px;

min-height: 100px;

background-color: #fff;

}

.jogo-item {

background-color: #f8f9fa;

padding: 10px;

margin: 5px 0;

border-radius: 4px;

display: flex;

justify-content: space-between;

align-items: center;

box-shadow: 0 1px 3px rgba(0, 0, 0, 0.1);

}

.jogo-numeros {

display: flex;

gap: 5px;

}

.jogo-numero {

background-color: var(--primary-color);

color: white;

width: 30px;

height: 30px;

border-radius: 50%;

display: flex;

align-items: center;

justify-content: center;

font-size: 14px;

font-weight: bold;

}

.btn-remover {

background-color: var(--danger-color);

color: white;

border: none;

border-radius: 4px;

padding: 5px 10px;

cursor: pointer;

font-size: 12px;

}

.btn-remover:hover {

opacity: 0.9;

}

.overlay {

position: fixed;

top: 0;

left: 0;

right: 0;

bottom: 0;

background-color: rgba(0, 0, 0, 0.7);

display: flex;

justify-content: center;

align-items: center;

z-index: 1000;

backdrop-filter: blur(5px);

}

.progress {

background-color: white;

padding: 20px;

border-radius: 8px;

width: 300px;

text-align: center;

box-shadow: 0 4px 6px rgba(0, 0, 0, 0.1);

}

.progress-bar {

height: 6px;

background-color: var(--primary-color);

border-radius: 3px;

margin: 10px 0;

position: relative;

overflow: hidden;

}

.progress-bar::after {

content: '';

position: absolute;

top: 0;

left: 0;

right: 0;

bottom: 0;

background: linear-gradient(

90deg,

transparent,

rgba(255, 255, 255, 0.4),

transparent

);

animation: shimmer 1s infinite;

}

.progress-text {

margin-top: 10px;

color: #495057;

font-weight: 500;

}

.detalhes-resultados {

margin-top: 20px;

}

.export-buttons {

display: flex;

gap: 10px;

margin: 10px 0;

justify-content: flex-end;

}

.hidden {

display: none !important;

}

.btn-export {

background-color: #008751;

color: white;

border: none;

padding: 8px 16px;

border-radius: 4px;

cursor: pointer;

font-size: 14px;

display: flex;

align-items: center;

gap: 5px;

transition: background-color 0.3s;

}

.btn-export:hover {

background-color: #006b41;

}

.btn-export i {

font-size: 16px;

}

@keyframes shimmer {

0% {

transform: translateX(-100%);

}

100% {

transform: translateX(100%);

}

}

@keyframes pulse {

0% {

box-shadow: 0 0 0 0 rgba(0, 135, 81, 0.4);

}

70% {

box-shadow: 0 0 0 10px rgba(0, 135, 81, 0);

}

100% {

box-shadow: 0 0 0 0 rgba(0, 135, 81, 0);

}

}

.sem-resultados {

text-align: center;

padding: 20px;

color: #666;

}

.resultado-item {

background-color: #fff;

padding: 20px;

margin-bottom: 15px;

border-radius: 8px;

box-shadow: 0 2px 4px rgba(0,0,0,0.1);

}

.resultado-numeros {

margin: 15px 0;

}

.numeros-lista {

display: flex;

flex-wrap: wrap;

gap: 8px;

margin: 10px 0;

}

.numero-sorteado,

.numero-jogado {

display: inline-flex;

align-items: center;

justify-content: center;

width: 35px;

height: 35px;

border-radius: 50%;

background: #f0f0f0;

font-weight: bold;

}

.numero-jogado.acerto {

background: #28a745;

color: white;

}

.acertos-info,

.premio-info {

margin: 10px 0;

font-size: 1.1em;

}

.premio-info {

color: #28a745;

}

.jogo-checkbox {

margin-right: 10px;

width: 18px;

height: 18px;

cursor: pointer;

}

.jogo-item {

display: flex;

align-items: center;

padding: 12px;

margin: 8px 0;

background-color: #f8f9fa;

border-radius: 6px;

border: 1px solid #e9ecef;

transition: all 0.2s ease;

}

.jogo-item:hover {

background-color: #e9ecef;

transform: translateY(-1px);

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);

}

.acoes-jogos {

display: flex;

gap: 10px;

margin: 15px 0;

padding: 10px;

background-color: #f8f9fa;

border-radius: 6px;

justify-content: flex-end;

}

.btn-acao {

padding: 8px 16px;

border: none;

border-radius: 4px;

cursor: pointer;

font-weight: 500;

transition: all 0.2s ease;

background-color: #6c757d;

color: white;

}

.btn-acao:disabled {

opacity: 0.5;

cursor: not-allowed;

}

.btn-acao:not(:disabled):hover {

opacity: 0.9;

transform: translateY(-1px);

}

.jogo-item.selecionado {

background-color: rgba(0, 135, 81, 0.1);

border-color: #008751;

}

#contador-jogos {

font-weight: bold;

color: var(--primary-color);

}

.valor-premio {

font-size: 0.9em;

color: #666;

margin-top: 5px;

font-weight: 500;

}

.premio-info strong {

color: #28a745;

}

.premio-info strong:only-child {

color: #666;

}

.btn-cancelar {

margin-top: 15px;

padding: 8px 16px;

background-color: #dc3545;

color: white;

border: none;

border-radius: 4px;

cursor: pointer;

font-weight: 500;

transition: all 0.2s ease;

}

.btn-cancelar:hover {

background-color: #c82333;

}

.tabela-resumo {

margin-top: 30px;

padding: 20px;

background-color: white;

border-radius: 8px;

box-shadow: 0 2px 4px rgba(0,0,0,0.1);

}

.tabela-resumo h2 {

color: #333;

margin-bottom: 20px;

text-align: center;

}

.tabela-container {

overflow-x: auto;

}

.tabela-premios {

width: 100%;

border-collapse: collapse;

margin: 20px 0;

font-size: 0.9em;

min-width: 800px;

background-color: white;

}

.tabela-premios thead tr {

background-color: #008751;

color: white;

text-align: left;

}

.tabela-premios th,

.tabela-premios td {

padding: 12px 15px;

border-bottom: 1px solid #dddddd;

}

.tabela-premios tbody tr:nth-of-type(even) {

background-color: #f3f3f3;

}

.tabela-premios tbody tr:hover {

background-color: #f5f5f5;

}

.numeros-tabela {

display: flex;

gap: 5px;

flex-wrap: wrap;

}

.numero-tabela {

width: 24px;

height: 24px;

border-radius: 50%;

background-color: #e9ecef;

display: flex;

align-items: center;

justify-content: center;

font-size: 12px;

}

.numero-tabela.acerto {

background-color: #28a745;

color: white;

}

.tabela-premios tfoot {

background-color: #f8f9fa;

font-weight: 500;

}

.tabela-premios tfoot td {

padding: 15px;

border-top: 2px solid #008751;

}

.total-premios {

color: #28a745;

font-weight: bold;

}

.jogos-mais-sorteados {

margin-top: 30px;

padding: 20px;

background-color: white;

border-radius: 8px;

box-shadow: 0 2px 4px rgba(0,0,0,0.1);

}

.jogos-mais-sorteados h2 {

color: #333;

margin-bottom: 20px;

text-align: center;

}

.distribuicao-badge {

display: inline-block;

background-color: #f0f0f0;

padding: 4px 8px;

border-radius: 12px;

margin: 2px;

font-size: 0.9em;

}

/\* Mantendo as variáveis de cores originais \*/

:root {

--primary-color: #008751;

--secondary-color: #005c36;

--danger-color: #dc3545;

--success-color: #28a745;

}

/\* Estilos base mantidos... \*/

/\* Adicionando Media Queries para responsividade \*/

/\* Ajustes para tablets e telas menores \*/

@media screen and (max-width: 1024px) {

.container {

padding: 15px;

margin: 10px;

}

.cards {

grid-template-columns: repeat(2, 1fr);

gap: 15px;

}

.numeros {

grid-template-columns: repeat(8, 1fr);

gap: 6px;

padding: 10px;

}

.numero {

width: 35px;

height: 35px;

font-size: 14px;

}

.controles {

flex-wrap: wrap;

gap: 15px;

}

.input-group {

flex: 1;

min-width: 150px;

}

.tabela-premios {

font-size: 0.85em;

min-width: 600px;

}

}

/\* Ajustes para smartphones \*/

@media screen and (max-width: 768px) {

body {

padding: 10px;

}

.container {

padding: 10px;

}

.volante {

padding: 10px;

margin: 10px 0;

}

.numeros {

grid-template-columns: repeat(6, 1fr);

gap: 5px;

}

.numero {

width: 32px;

height: 32px;

font-size: 13px;

}

.cards {

grid-template-columns: 1fr;

gap: 10px;

}

.acoes {

flex-wrap: wrap;

gap: 8px;

}

.btn {

flex: 1;

min-width: 120px;

font-size: 14px;

padding: 8px 15px;

}

.drop-zone {

height: 150px;

padding: 15px;

}

.drop-zone-content p {

font-size: 0.9em;

}

.jogo-item {

flex-wrap: wrap;

gap: 10px;

}

.jogo-numeros {

order: 2;

width: 100%;

justify-content: center;

}

.btn-remover {

order: 3;

width: 100%;

}

.jogo-checkbox {

order: 1;

}

/\* Ajustes para tabelas em telas pequenas \*/

.tabela-container {

margin: 0 -10px;

padding: 0 10px;

}

.tabela-premios {

font-size: 0.8em;

min-width: 450px;

}

.tabela-premios th,

.tabela-premios td {

padding: 8px 10px;

}

.numero-tabela {

width: 20px;

height: 20px;

font-size: 11px;

}

.distribuicao-badge {

font-size: 0.8em;

padding: 3px 6px;

}

/\* Ajustes para os resultados \*/

.resultado-item {

padding: 15px;

}

.numero-sorteado,

.numero-jogado {

width: 28px;

height: 28px;

font-size: 12px;

}

/\* Ajustes para botões de ação \*/

.acoes-jogos {

flex-direction: column;

gap: 8px;

}

.btn-acao {

width: 100%;

}

/\* Overlay e progresso \*/

.progress {

width: 90%;

max-width: 280px;

}

}

/\* Ajustes para telas muito pequenas \*/

@media screen and (max-width: 375px) {

.numeros {

grid-template-columns: repeat(5, 1fr);

gap: 4px;

}

.numero {

width: 30px;

height: 30px;

font-size: 12px;

}

.input-group input {

width: 100%;

}

.controles {

flex-direction: column;

align-items: stretch;

}

.checkbox-group {

justify-content: center;

}

.tabela-premios {

min-width: 300px;

font-size: 0.75em;

}

.numeros-tabela {

gap: 3px;

}

.numero-tabela {

width: 18px;

height: 18px;

font-size: 10px;

}

}

/\* Orientação paisagem para dispositivos móveis \*/

@media screen and (max-width: 768px) and (orientation: landscape) {

.numeros {

grid-template-columns: repeat(10, 1fr);

}

.cards {

grid-template-columns: repeat(3, 1fr);

}

.volante {

margin: 10px 0;

}

}