server

import os

import requests

from flask import Flask, jsonify, request, send\_from\_directory

from flask\_cors import CORS

from datetime import datetime, timedelta, timezone

import calendar

from dateutil.relativedelta import relativedelta

# --- CONFIGURAÇÃO ---

OPENMETEO\_FORECAST\_URL = "https://api.open-meteo.com/v1/forecast"

OPENMETEO\_HISTORICAL\_URL = "https://archive-api.open-meteo.com/v1/era5"

# 🛑 ALTERADO: Mapeamento de URLs de câmeras públicas (Incluindo São José dos Campos)

CAMERA\_URLS = {

"São Paulo": "https://climaaovivo.com.br/sp/sao-paulo/alpha-sat-paulista-wall-street-leste",

"Rio de Janeiro": "https://climaaovivo.com.br/rj/rio-de-janeiro/rio-de-janeiro-samba-hoteis-bossa-nova-ipanema",

"Recife": "https://climaaovivo.com.br/pe/recife/recife-bugan-hotel-by-atlantica",

"Belo Horizonte": "https://climaaovivo.com.br/mg/belo-horizonte/luxemburgo-e-santo-antonio",

"Curitiba": "https://climaaovivo.com.br/pr/curitiba/curitiba-cemiterio-vertical-de-curitiba",

"Fortaleza": "https://climaaovivo.com.br/ce/abaiara-torre-da-linkcariri",

"Belém": "https://climaaovivo.com.br/pa/belem",

"Goiânia": "https://climaaovivo.com.br/go/goiania/goiania-golden-tulip-goiania-address",

"São José dos Campos": "https://climaaovivo.com.br/sp/sao-jose-dos-campos/sao-jose-dos-campos-hotel-golden-tulip", # 🛑 LINK ADICIONADO

}

# LISTA DE CAPITAIS PARA MONITORAMENTO (Campinas alterada para São José dos Campos)

CAPITAIS\_BRASIL = [

{"nome": "São Paulo", "estado": "SP", "lat": -23.55, "lon": -46.63},

{"nome": "Rio de Janeiro", "estado": "RJ", "lat": -22.90, "lon": -43.20},

{"nome": "Belo Horizonte", "estado": "MG", "lat": -19.91, "lon": -43.93},

{"nome": "Salvador", "estado": "BA", "lat": -12.97, "lon": -38.50},

{"nome": "Brasília", "estado": "DF", "lat": -15.78, "lon": -47.92},

{"nome": "Curitiba", "estado": "PR", "lat": -25.42, "lon": -49.27},

{"nome": "Recife", "estado": "PE", "lat": -8.05, "lon": -34.88},

{"nome": "Fortaleza", "estado": "CE", "lat": -3.73, "lon": -38.52},

{"nome": "Manaus", "estado": "AM", "lat": -3.11, "lon": -60.02},

{"nome": "Porto Alegre", "estado": "RS", "lat": -30.03, "lon": -51.23},

{"nome": "Belém", "estado": "PA", "lat": -1.45, "lon": -48.50},

{"nome": "Goiânia", "estado": "GO", "lat": -16.68, "lon": -49.25},

{"nome": "São José dos Campos", "estado": "SP", "lat": -23.1794, "lon": -45.8872}, # Alterada

]

# LISTA DE CIDADES COM RISCO HISTÓRICO DE ESCORREGAMENTO

CIDADES\_RISCO\_MONITORADAS = [

{"nome": "Petrópolis", "estado": "RJ", "lat": -22.505, "lon": -43.18},

{"nome": "Teresópolis", "estado": "RJ", "lat": -22.412, "lon": -42.966},

{"nome": "Nova Friburgo", "estado": "RJ", "lat": -22.281, "lon": -42.531},

{"nome": "Angra dos Reis", "estado": "RJ", "lat": -23.006, "lon": -44.318},

{"nome": "Ubatuba", "estado": "SP", "lat": -23.433, "lon": -45.083},

{"nome": "Caraguatatuba", "estado": "SP", "lat": -23.621, "lon": -45.413},

{"nome": "São Sebastião", "estado": "SP", "lat": -23.760, "lon": -45.409},

{"nome": "Guarujá", "estado": "SP", "lat": -23.993, "lon": -46.256},

{"nome": "Franco da Rocha", "estado": "SP", "lat": -23.327, "lon": -46.725},

{"nome": "Mauá", "estado": "SP", "lat": -23.667, "lon": -46.461},

{"nome": "Ouro Preto", "estado": "MG", "lat": -20.385, "lon": -43.504},

{"nome": "Jaboatão dos Guararapes", "estado": "PE", "lat": -8.113, "lon": -35.015},

{"nome": "Olinda", "estado": "PE", "lat": -8.008, "lon": -34.855},

{"nome": "Camaragibe", "estado": "PE", "lat": -8.023, "lon": -34.984},

{"nome": "Maceió", "estado": "AL", "lat": -9.665, "lon": -35.735},

{"nome": "Blumenau", "estado": "SC", "lat": -26.919, "lon": -49.066},

{"nome": "Florianópolis", "estado": "SC", "lat": -27.596, "lon": -48.549},

{"nome": "Muçum", "estado": "RS", "lat": -29.165, "lon": -51.868},

]

app = Flask(\_\_name\_\_, static\_folder='web', static\_url\_path='')

CORS(app)

@app.route('/')

def serve\_index():

return send\_from\_directory('web', 'welcome.html')

# Função auxiliar para conversão de código do tempo

def converter\_codigo\_tempo(code):

codes = {

0: "Céu Limpo", 1: "Céu Parcialmente Nublado", 2: "Céu Nublado", 3: "Céu Encoberto",

45: "Neblina", 48: "Névoa", 51: "Chuvisco Leve", 53: "Chuvisco Moderado", 55: "Chuvisco Intenso",

61: "Chuva Leve", 63: "Chuva Moderada", 65: "Chuva Forte", 80: "Pancadas de Chuva Leve",

81: "Pancadas de Chuva Moderada", 82: "Pancadas de Chuva Forte", 95: "Tempestade", 96: "Tempestade com Granizo Leve", 99: "Tempestade com Granizo Forte"

}

return codes.get(code, f"Cód. {code} (N/D)")

# Função para determinar o Nível de Risco

def determinar\_nivel(valor):

if valor >= 30: return {"nivel": "VERMELHO", "cor": "#FF0000"}

if valor >= 20: return {"nivel": "LARANJA", "cor": "#FFA500"}

if valor >= 10: return {"nivel": "AMARELO", "cor": "#FFFF00"}

return {"nivel": "VERDE", "cor": "#008000"}

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

def get\_capitais\_list():

return jsonify(CAPITAIS\_BRASIL)

@app.route('/api/cidades\_risco', methods=['GET'])

def get\_cidades\_risco\_list():

return jsonify(CIDADES\_RISCO\_MONITORADAS)

# =========================================================

# ENDPOINT PARA SALA DE MONITORAMENTO (CARDS)

# =========================================================

@app.route('/api/capitais\_risco', methods=['GET'])

def get\_capitais\_risco():

dados\_monitoramento = []

agora\_utc = datetime.now(timezone.utc)

end\_time\_hist = agora\_utc - timedelta(hours=96)

start\_time\_hist = end\_time\_hist - timedelta(hours=72)

start\_date\_hist = start\_time\_hist.strftime('%Y-%m-%d')

end\_date\_hist = end\_time\_hist.strftime('%Y-%m-%d')

for capital in CAPITAIS\_BRASIL:

lat, lon = capital['lat'], capital['lon']

nome\_capital = capital['nome']

chuva\_futura, chuva\_historica, maior\_risco = 0, 0, 0

nivel\_risco = {"nivel": "ERRO", "cor": "#999999"}

try:

params\_forecast = {"latitude": lat, "longitude": lon, "hourly": "precipitation", "forecast\_days": 4, "timezone": "auto"}

resp\_forecast = requests.get(OPENMETEO\_FORECAST\_URL, params=params\_forecast)

resp\_forecast.raise\_for\_status()

chuva\_futura = sum(p for p in resp\_forecast.json().get('hourly', {}).get('precipitation', [])[:72] if p is not None)

params\_chuva\_hist = {"latitude": lat, "longitude": lon, "start\_date": start\_date\_hist, "end\_date": end\_date\_hist, "hourly": "precipitation", "timezone": "auto", "models": "era5"}

resp\_chuva\_hist = requests.get(OPENMETEO\_HISTORICAL\_URL, params=params\_chuva\_hist)

resp\_chuva\_hist.raise\_for\_status()

dados\_chuva\_hist = resp\_chuva\_hist.json()

chuva\_historica = sum(p for p in dados\_chuva\_hist['hourly']['precipitation'] if p is not None)

maior\_risco = max(chuva\_historica, chuva\_futura)

nivel\_risco = determinar\_nivel(maior\_risco)

# 🛑 CORRIGIDO: Adiciona a URL da câmera, que será None se a cidade não estiver no CAMERA\_URLS

camera\_url = CAMERA\_URLS.get(nome\_capital)

dados\_monitoramento.append({

"capital": nome\_capital,

"estado": capital['estado'],

"risco\_nivel": nivel\_risco['nivel'],

"maior\_risco\_valor": maior\_risco,

"camera\_url": camera\_url

})

except Exception as e:

print(f"[{datetime.now().isoformat()}] Erro para {nome\_capital}: {e}")

dados\_monitoramento.append({"capital": nome\_capital, "estado": capital['estado'], "risco\_nivel": "ERRO", "maior\_risco\_valor": 0})

return jsonify(dados\_monitoramento)

# =========================================================

# ENDPOINT PARA DADOS DO PAINEL LATERAL

# =========================================================

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

def get\_weather\_data():

lat, lon = request.args.get('lat'), request.args.get('lon')

nome\_cidade = request.args.get('nome\_cidade', 'Local')

if not lat or not lon: return jsonify({"error": "Lat e Lon são obrigatórios."}), 400

try:

end\_hist = datetime.now(timezone.utc) - timedelta(hours=96)

start\_hist = end\_hist - timedelta(hours=72)

params\_hist = {"latitude": lat, "longitude": lon, "start\_date": start\_hist.strftime('%Y-%m-%d'), "end\_date": end\_hist.strftime('%Y-%m-%d'), "hourly": "precipitation", "timezone": "auto", "models": "era5"}

resp\_hist = requests.get(OPENMETEO\_HISTORICAL\_URL, params=params\_hist)

resp\_hist.raise\_for\_status()

chuva\_hist = sum(p for p in resp\_hist.json().get('hourly', {}).get('precipitation', []) if p is not None)

params\_forecast = {"latitude": lat, "longitude": lon, "hourly": "temperature\_2m,apparent\_temperature,windspeed\_10m,windgusts\_10m,surface\_pressure,weather\_code,precipitation,relative\_humidity\_2m,dewpoint\_2m", "forecast\_days": 4, "timezone": "auto"}

resp\_forecast = requests.get(OPENMETEO\_FORECAST\_URL, params=params\_forecast)

resp\_forecast.raise\_for\_status()

hourly = resp\_forecast.json().get('hourly', {})

def get\_val(key):

return hourly.get(key, [None])[0]

chuva\_fut = sum(p for p in hourly.get('precipitation', [])[:72] if p is not None)

return jsonify({

"temperatura": get\_val('temperature\_2m') or 0,

"cidade\_nome": nome\_cidade,

"chuva\_72h\_hist": chuva\_hist,

"chuva\_72h\_fut": chuva\_fut,

"sensacao\_termica": get\_val('apparent\_temperature'),

"velocidade\_vento": get\_val('windspeed\_10m'),

"pressao": get\_val('surface\_pressure'),

"descricao\_tempo": converter\_codigo\_tempo(get\_val('weather\_code')),

"umidade\_relativa": get\_val('relative\_humidity\_2m'),

"ponto\_orvalho": get\_val('dewpoint\_2m'),

"rajada\_vento": get\_val('windgusts\_10m')

})

except Exception as e:

print(f"Erro na rota /api/weather: {e}")

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

# =========================================================

# ENDPOINTS PARA OS GRÁFICOS DO DASHBOARD

# =========================================================

@app.route('/api/historical\_pluvio', methods=['GET'])

def get\_historical\_pluvio\_data():

lat, lon = request.args.get('lat'), request.args.get('lon')

periodo\_horas = int(request.args.get('periodo', 72))

nome\_cidade\_frontend = request.args.get('nome\_cidade', 'Local')

if not lat or not lon: return jsonify({"error": "Lat e Lon são obrigatórios."}), 400

end\_time\_utc = datetime.now(timezone.utc) - timedelta(days=5)

start\_time\_utc = end\_time\_utc - timedelta(hours=periodo\_horas)

params = {"latitude": lat, "longitude": lon, "start\_date": start\_time\_utc.strftime('%Y-%m-%d'), "end\_date": end\_time\_utc.strftime('%Y-%m-%d'), "hourly": "precipitation", "timezone": "auto", "models": "era5"}

try:

resp = requests.get(OPENMETEO\_HISTORICAL\_URL, params=params)

resp.raise\_for\_status()

data = resp.json()

precip = data.get('hourly', {}).get('precipitation', [])

return jsonify({"cidade": nome\_cidade\_frontend, "municipio": nome\_cidade\_frontend, "volume\_pluviometria": precip, "data\_pluviometria": data.get('hourly', {}).get('time', []), "acumulado\_total": sum(p for p in precip if p is not None)})

except Exception as e:

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

@app.route('/api/forecast\_chart', methods=['GET'])

def get\_forecast\_chart\_data():

lat = request.args.get('lat')

lon = request.args.get('lon')

nome\_cidade\_frontend = request.args.get('nome\_cidade', 'Local')

if not lat or not lon: return jsonify({"error": "Lat e Lon são obrigatórios."}), 400

try:

params\_forecast = {"latitude": lat, "longitude": lon, "hourly": "temperature\_2m,apparent\_temperature,precipitation\_probability,precipitation,dewpoint\_2m,relative\_humidity\_2m,windspeed\_10m,windgusts\_10m,surface\_pressure,weather\_code", "forecast\_days": 7, "timezone": "auto"}

resp\_forecast = requests.get(OPENMETEO\_FORECAST\_URL, params=params\_forecast)

resp\_forecast.raise\_for\_status()

dados\_forecast = resp\_forecast.json()

dados\_forecast['cidade\_nome'] = nome\_cidade\_frontend

dados\_forecast['municipio'] = nome\_cidade\_frontend

return jsonify(dados\_forecast)

except Exception as e:

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

# --- Bloco de execução para desenvolvimento local ---

# Este bloco SÓ é executado quando você roda `python server.py`

# O Gunicorn (servidor de produção) ignora este bloco completamente.

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

app.run(debug=True)

Index

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

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

<title>RiskGeo Solutions - Monitoramento Climático</title>

<script src="https://cdn.jsdelivr.net/npm/chart.js@4.4.0/dist/chart.umd.min.js"></script>

<link rel="stylesheet" href="https://unpkg.com/leaflet/dist/leaflet.css" />

<link rel="stylesheet" href="https://unpkg.com/leaflet-control-geocoder/dist/Control.Geocoder.css" />

<link rel="stylesheet" href="style.css">

<style>

:root {

--cor-faixa-escura: rgba(0, 50, 150, 0.85);

--cor-texto-claro: white;

--cor-texto-escuro: #003296;

--cor-fundo-dashboard: #F0F8FF;

}

body { margin: 0; padding: 0; font-family: 'Arial', sans-serif; }

#map-container, #dashboard-container, #monitor-room-container { position: absolute; top: 0; left: 0; width: 100%; height: 100vh; transition: opacity 0.5s; }

#map-container { z-index: 100; }

#map { width: 100%; height: 100%; }

.app-header { position: absolute; top: 0; left: 0; right: 0; height: 75px; background-color: var(--cor-faixa-escura); color: var(--cor-texto-claro); z-index: 1001; padding: 0 25px; display: flex; flex-direction: column; justify-content: center; align-items: flex-start; }

.header-top-row { width: 100%; display: flex; justify-content: space-between; align-items: center; padding-bottom: 5px; }

.app-logo { font-size: 1.8em; font-weight: 700; letter-spacing: 1px; color: var(--cor-texto-claro); text-decoration: none; cursor: pointer; }

.aviso-legal { font-size: 0.75em; color: #FFD700; width: 100%; text-align: left; margin-top: -5px; }

.header-links-group { display: flex; align-items: center; gap: 15px; }

.header-link, .header-link:visited { color: var(--cor-texto-claro); text-decoration: none; font-size: 1.1em; padding: 7px 12px; border: 1px solid var(--cor-texto-claro); border-radius: 4px; transition: background-color 0.3s; }

.header-link:hover { background-color: rgba(255, 255, 255, 0.2); }

.leaflet-control-geocoder { position: absolute; top: 85px !important; left: 10px !important; z-index: 999; width: 245px !important; max-width: 300px !important; }

.leaflet-control-geocoder input { width: 100% !important; font-size: 22px !important; padding-top: 8px !important; padding-bottom: 8px !important; background-color: #000000 !important; color: white; border: 1px solid white; }

.leaflet-control-geocoder input::placeholder { color: #ccc; opacity: 1; }

.data-panel { position: absolute; top: 85px; right: 20px; z-index: 1000; background-color: white; padding: 15px; border-radius: 8px; box-shadow: 0 4px 8px rgba(0,0,0,0.5); font-family: inherit; width: 280px; border: none; transition: background-color 0.5s ease; }

.color-dark { color: #333; }

.color-light { color: white; }

.data-panel h2, #nome-cidade, .data-panel p, .data-panel .value { color: inherit !important; }

#map-footer-note { position: absolute; bottom: 10px; left: 50%; transform: translateX(-50%); z-index: 1000; background: rgba(0, 0, 0, 0.6); color: white; padding: 5px 15px; border-radius: 5px; font-size: 0.8em; text-align: center; }

#monitor-room-btn { padding: 7px 12px; background-color: #3f51b5; color: white; border: 1px solid white; border-radius: 4px; cursor: pointer; box-shadow: 0 2px 5px rgba(0,0,0,0.2); font-weight: bold; font-size: 1.1em; transition: background-color 0.3s; }

#monitor-room-btn:hover { background-color: #303f9f; }

#monitor-room-container { background-color: #f4f4f4; padding: 20px; overflow-y: auto; display: none; padding-top: 95px; }

#monitor-room-content { width: 95%; max-width: 1400px; margin: 0 auto; padding-top: 20px; }

.monitor-header { display: flex; justify-content: flex-end; align-items: center; margin-bottom: 20px; }

#monitor-footer-note { text-align: center; font-size: 0.9em; color: #555; padding: 15px 0; margin-top: 30px; border-top: 1px solid #ccc; }

.nivel-VERDE { background-color: #008000; color: white; }

.nivel-AMARELO { background-color: #FFFF00; color: black; }

.nivel-LARANJA { background-color: #FFA500; color: white; }

.nivel-VERMELHO { background-color: #FF0000; color: white; }

.nivel-ERRO { background-color: #999999; color: white; }

.loading-row { text-align: center; font-style: italic; color: #666; font-size: 1.2em; padding: 20px; }

#back-to-map-from-monitor-btn { padding: 10px 20px; background-color: #007bff; color: white; border: none; border-radius: 5px; cursor: pointer; font-size: 1em; transition: background-color 0.3s; }

#back-to-map-from-monitor-btn:hover { background-color: #0056b3; }

#dashboard-container { display: none; padding-top: 95px; }

#dashboard-content { width: 95%; max-width: 1400px; margin: 0 auto; }

.charts-container { display: flex; flex-wrap: wrap; justify-content: space-around; gap: 20px; margin-top: 20px; }

#pluviometria-card, #previsao-card { position: relative; flex: 1; min-width: 400px; height: 45vh; padding: 15px; background: white; box-shadow: 0 2px 5px rgba(0,0,0,0.1); border-radius: 8px; }

#capitais-grid-container { display: grid; grid-template-columns: repeat(auto-fill, minmax(300px, 1fr)); gap: 20px; }

.capital-card { background: white; border-radius: 8px; box-shadow: 0 4px 8px rgba(0,0,0,0.1); padding: 20px; display: flex; justify-content: space-between; align-items: center; transition: transform 0.2s ease, box-shadow 0.2s ease; }

.capital-card:hover { transform: translateY(-5px); box-shadow: 0 6px 12px rgba(0,0,0,0.15); }

.card-city-name { font-size: 1.2em; font-weight: 600; color: #333; }

.card-risk-level { padding: 8px 15px; border-radius: 5px; font-weight: bold; font-size: 0.9em; text-align: center; min-width: 110px; }

.capital-marker-icon { background-color: #fff; border: 2px solid white; border-radius: 50%; box-shadow: 0 2px 5px rgba(0,0,0,0.4); color: white; font-size: 14px; font-weight: bold; text-align: center; line-height: 26px; width: 30px; height: 30px; }

.dashboard-header { position: relative; text-align: center; padding: 10px 0; margin-bottom: 20px; border-bottom: 1px solid #ddd; }

#back-to-map-btn { position: absolute; right: 0; top: 50%; transform: translateY(-50%); padding: 10px 20px; background-color: #007bff; color: white; border: none; border-radius: 5px; cursor: pointer; transition: background-color 0.3s; }

#back-to-map-btn:hover { background-color: #0056b3; }

.seletor-container { display: flex; flex-wrap: wrap; align-items: center; gap: 15px; padding: 15px; background-color: #f8f9fa; border-radius: 8px; margin-bottom: 20px; font-size: 1.1em; }

.dashboard-details-grid { display: grid; grid-template-columns: repeat(auto-fill, minmax(200px, 1fr)); gap: 20px; margin-top: 20px; }

.dashboard-details-grid .card { background: white; padding: 20px; border-radius: 8px; box-shadow: 0 4px 8px rgba(0,0,0,0.05); text-align: center; border: 1px solid #eee; }

.dashboard-details-grid .card h4 { margin: 0 0 10px 0; font-size: 0.9em; color: #555; font-weight: normal; }

.dashboard-details-grid .card .large-value { margin: 0; font-size: 1.5em; font-weight: bold; color: var(--cor-texto-escuro); }

</style>

</head>

<body>

<header class="app-header">

<div class="header-top-row">

<a href="https://www.riskgeo.com.br" target="\_blank" class="app-logo">RiskGeo Solutions</a>

<div class="header-links-group">

<button id="monitor-room-btn">Sala de Monitoramento</button>

<a href="https://www.riskgeo.com.br" target="\_blank" class="header-link">Site</a>

</div>

</div>

<span class="aviso-legal">AVISO: Dados para fins demonstrativos. Não utilizar como ferramenta de risco operacional.</span>

</header>

<div id="map-container">

<div id="map"></div>

<div id="alerta-panel" class="data-panel">

<h2 id="alerta-titulo"></h2>

<p id="nome-cidade">Carregando Local...</p>

<p>Chuva Acumulada (72h): <span id="chuva-agora" class="value">Aguardando...</span></p>

<p>Previsão Chuva (72h): <span id="chuva-futura" class="value">Aguardando...</span></p>

<p>Temperatura Atual: <span id="temperatura" class="value">Aguardando...</span></p>

<button id="open-dashboard-btn" style="width:100%; margin-top:10px; padding: 8px;">Ver Gráfico de Pluviometria</button>

</div>

<div id="map-footer-note">

NOTA: As faixas de níveis de risco atuais foram alteradas unicamente para a verificação das funcionalidades da aplicação.

</div>

</div>

<div id="monitor-room-container" style="display: none;">

<div id="monitor-room-content">

<header class="monitor-header">

<button id="back-to-map-from-monitor-btn">← Voltar ao Mapa</button>

</header>

<div id="capitais-grid-container">

<div class="loading-row">Aguardando dados das cidades...</div>

</div>

<div id="monitor-footer-note">

NOTA: As faixas de níveis de risco atuais foram alteradas unicamente para a verificação das funcionalidades da aplicação.

</div>

</div>

</div>

<div id="dashboard-container" style="display: none;">

<div id="dashboard-content">

<header class="dashboard-header">

<h1 id="dashboard-title">Pluviometria Histórica para [Cidade]</h1>

<button id="back-to-map-btn">← Voltar ao Mapa</button>

</header>

<div class="seletor-container">

<label for="periodo-selector">Período Histórico:</label>

<select id="periodo-selector">

<option value="24">24 horas</option>

<option value="48">48 horas</option>

<option value="72" selected>72 horas</option>

<option value="96">96 horas</option>

</select>

<span id="municipio-display"></span>

<button id="toggle-detalhes-btn" style="padding: 8px 15px; background-color: #28a745; color: white; border: none; border-radius: 4px; cursor: pointer;">Ver Detalhes Adicionais</button>

</div>

<div id="detalhes-adicionais-container" style="display: none;">

<h2 style="margin-top: 0; color: #003296;">Informações Detalhadas (Atual)</h2>

<div class="dashboard-details-grid">

<div class="card"><h4>Condição Atual</h4><p id="detail-descricao" class="large-value">--</p></div>

<div class="card"><h4>Sensação Térmica</h4><p id="detail-feels-like" class="large-value">--</p></div>

<div class="card"><h4>Umidade Relativa</h4><p id="detail-humidity" class="large-value">--</p></div>

<div class="card"><h4>Ponto de Orvalho</h4><p id="detail-dewpoint" class="large-value">--</p></div>

<div class="card"><h4>Pressão Atmosférica</h4><p id="detail-pressure" class="large-value">--</p></div>

<div class="card"><h4>Vento Atual (10m)</h4><p id="detail-wind" class="large-value">--</p></div>

<div class="card"><h4>Rajada de Vento (Máx)</h4><p id="detail-gusts" class="large-value">--</p></div>

</div>

</div>

<div class="charts-container">

<div id="pluviometria-card">

<h2 style="text-align:center;">Pluviometria Acumulada (Histórico)</h2>

<canvas id="pluvioChart"></canvas>

</div>

<div id="previsao-card">

<h2 style="text-align:center;">Pluviometria Acumulada (Previsão)</h2>

<canvas id="previsaoChart"></canvas>

</div>

</div>

<div id="monthlyChartContainer" style="display:none;"><h2 style="text-align:center;">Histórico Anual: Temperatura Média e Precipitação Total (Mensal)</h2><canvas id="monthlyChart"></canvas></div>

</div>

</div>

<script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>

<script src="https://unpkg.com/leaflet-control-geocoder/dist/Control.Geocoder.js"></script>

<script src="script.js"></script>

</body>

</html>

Script - // Constantes e Variáveis Globais

const INITIAL\_LATITUDE = -15.78; // Latitude de Brasília (centro do mapa)

const INITIAL\_LONGITUDE = -47.92; // Longitude de Brasília (centro do mapa)

const ZOOM\_LEVEL = 4; // Nível de zoom para ver o Brasil

let updateIntervalId = null;

let currentCoords = { lat: INITIAL\_LATITUDE, lon: INITIAL\_LONGITUDE };

let currentCityName = "Brasília"; // Cidade inicial para carregar os dados

let currentCharts = {};

let pluvioChartInstance = null;

// Mapeamento de níveis de risco para texto

const riskTextMap = {

VERDE: "Observação",

AMARELO: "Atenção",

LARANJA: "Alto Risco",

VERMELHO: "Risco Extremo",

ERRO: "Erro"

};

// Mapeamento de cor para Nível (para o painel lateral)

const colorToRiskMap = {

"#008000": "Observação", // Verde

"#FFFF00": "Atenção", // Amarelo

"#FFA500": "Alto Risco", // Laranja

"#FF0000": "Risco Extremo",// Vermelho

"#999": "Carregando"

};

// ==========================================================

// Mapeamento de Elementos e Navegação

// ==========================================================

const tempElement = document.getElementById('temperatura');

const chuvaHistElement = document.getElementById('chuva-agora');

const chuvaFutElement = document.getElementById('chuva-futura');

const nomeCidadeElement = document.getElementById('nome-cidade');

const alertaPanel = document.getElementById('alerta-panel');

const mapContainer = document.getElementById('map-container');

const dashboardContainer = document.getElementById('dashboard-container');

const dashboardTitle = document.getElementById('dashboard-title');

const openDashboardBtn = document.getElementById('open-dashboard-btn');

const backToMapBtn = document.getElementById('back-to-map-btn');

const periodoSelector = document.getElementById('periodo-selector');

const municipioDisplay = document.getElementById('municipio-display');

const toggleDetalhesBtn = document.getElementById('toggle-detalhes-btn');

const detalhesContainer = document.getElementById('detalhes-adicionais-container');

const monitorRoomBtn = document.getElementById('monitor-room-btn');

const monitorRoomContainer = document.getElementById('monitor-room-container');

const backToMapFromMonitorBtn = document.getElementById('back-to-map-from-monitor-btn');

const capitaisGridContainer = document.getElementById('capitais-grid-container');

const alertaTituloElement = document.getElementById('alerta-titulo');

// Mapeamento dos Cards de Detalhe

const detailDescricao = document.getElementById('detail-descricao');

const detailFeelsLike = document.getElementById('detail-feels-like');

const detailHumidity = document.getElementById('detail-humidity');

const detailDewpoint = document.getElementById('detail-dewpoint');

const detailPressure = document.getElementById('detail-pressure');

const detailWind = document.getElementById('detail-wind');

const detailGusts = document.getElementById('detail-gusts');

// ==========================================================

// FUNÇÕES DE NAVEGAÇÃO E SALA DE MONITORAMENTO

// ==========================================================

function showDashboard() {

mapContainer.style.display = 'none';

monitorRoomContainer.style.display = 'none';

dashboardContainer.style.display = 'block';

dashboardTitle.textContent = `Pluviometria Histórica para ${currentCityName}`;

municipioDisplay.textContent = `Município: ${currentCityName}/SP - Estação: Modelo ERA5`;

periodoSelector.value = '72';

fetchAndRenderPluvioChart(currentCoords.lat, currentCoords.lon, currentCityName, periodoSelector.value);

fetchAndRenderForecastChart(currentCoords.lat, currentCoords.lon, currentCityName);

fetchAndPreFillDetails(currentCoords.lat, currentCoords.lon, currentCityName);

map.invalidateSize();

}

function showMap() {

dashboardContainer.style.display = 'none';

monitorRoomContainer.style.display = 'none';

mapContainer.style.display = 'block';

detalhesContainer.style.display = 'none';

toggleDetalhesBtn.textContent = 'Ver Detalhes Adicionais';

map.invalidateSize();

}

function showMonitorRoom() {

mapContainer.style.display = 'none';

dashboardContainer.style.display = 'none';

monitorRoomContainer.style.display = 'block';

fetchCapitaisRisco();

}

async function fetchCapitaisRisco() {

capitaisGridContainer.innerHTML = '<div class="loading-row">Aguardando dados das cidades...</div>';

const API\_CAPITAIS\_URL = `/api/capitais\_risco`;

try {

const response = await fetch(API\_CAPITAIS\_URL);

const dados = await response.json();

if (!response.ok || dados.error) {

console.error('Erro ao buscar dados das capitais:', dados.error);

capitaisGridContainer.innerHTML = `<div class="loading-row">ERRO: ${dados.error || 'Falha na comunicação com o servidor.'}</div>`;

return;

}

renderCapitaisCards(dados);

} catch (error) {

console.error("Falha na rede ao buscar Sala de Monitoramento:", error);

capitaisGridContainer.innerHTML = `<div class="loading-row">ERRO DE CONEXÃO: Verifique o console do navegador e o terminal do servidor Flask.</div>`;

}

}

function renderCapitaisCards(capitais) {

capitaisGridContainer.innerHTML = '';

let htmlContent = '';

capitais.forEach(cidade => {

const riscoNivel = cidade.risco\_nivel;

const riscoTexto = riskTextMap[riscoNivel] || 'Indefinido'; // Pega o texto do mapa

// 🛑 NOVO: Botão de Câmera (mantido como no ponto de salvamento 2)

const cameraButton = cidade.camera\_url ?

`<a href="${cidade.camera\_url}" target="\_blank" class="camera-btn" style="background-color: #3f51b5; color: white; padding: 5px 10px; border-radius: 4px; text-decoration: none; font-size: 0.9em; margin-top: 10px;">Ver Câmera</a>` :

'';

htmlContent += `

<div class="capital-card" style="flex-direction: column; align-items: flex-start;">

<span class="card-city-name">${cidade.capital} (${cidade.estado})</span>

<span class="card-risk-level nivel-${riscoNivel}" title="Acumulado: ${cidade.maior\_risco\_valor !== undefined ? cidade.maior\_risco\_valor.toFixed(1) : '--'} mm" style="margin-top: 5px; margin-bottom: 5px;">

${riscoTexto}

</span>

${cameraButton}

</div>

`;

});

capitaisGridContainer.innerHTML = htmlContent;

}

const toggleDetalhes = () => {

const isHidden = detalhesContainer.style.display === 'none';

detalhesContainer.style.display = isHidden ? 'grid' : 'none';

toggleDetalhesBtn.textContent = isHidden ? 'Ocultar Detalhes Adicionais' : 'Ver Detalhes Adicionais';

};

periodoSelector.onchange = function() {

fetchAndRenderPluvioChart(currentCoords.lat, currentCoords.lon, currentCityName, this.value);

};

openDashboardBtn.onclick = showDashboard;

backToMapBtn.onclick = showMap;

toggleDetalhesBtn.onclick = toggleDetalhes;

monitorRoomBtn.onclick = showMonitorRoom;

backToMapFromMonitorBtn.onclick = showMap;

// ==========================================================

// FUNÇÕES DE BUSCA DE DADOS

// ==========================================================

function preencherCardsDetalhe(dados) {

detailDescricao.textContent = dados.descricao\_tempo || '--';

detailFeelsLike.textContent = (dados.sensacao\_termica !== undefined && dados.sensacao\_termica !== null) ? `${dados.sensacao\_termica.toFixed(1)} °C` : '--';

detailHumidity.textContent = (dados.umidade\_relativa !== undefined && dados.umidade\_relativa !== null) ? `${dados.umidade\_relativa.toFixed(0)} %` : '--';

detailDewpoint.textContent = (dados.ponto\_orvalho !== undefined && dados.ponto\_orvalho !== null) ? `${dados.ponto\_orvalho.toFixed(1)} °C` : '--';

detailPressure.textContent = (dados.pressao !== undefined && dados.pressao !== null) ? `${dados.pressao.toFixed(0)} hPa` : '--';

detailWind.textContent = (dados.velocidade\_vento !== undefined && dados.velocidade\_vento !== null) ? `${dados.velocidade\_vento.toFixed(1)} km/h` : '--';

detailGusts.textContent = (dados.rajada\_vento !== undefined && dados.rajada\_vento !== null) ? `${dados.rajada\_vento.toFixed(1)} km/h` : '--';

}

async function fetchAndPreFillDetails(latitude, longitude, nomeLocal) {

const API\_URL = `/api/weather?lat=${latitude}&lon=${longitude}&nome\_cidade=${encodeURIComponent(nomeLocal)}`;

try {

const response = await fetch(API\_URL);

const dados = await response.json();

if (!response.ok || dados.error) {

console.error('Falha ao pré-carregar detalhes:', dados.error);

return;

}

preencherCardsDetalhe(dados);

} catch (error) {

console.error("Falha na rede ao pré-carregar detalhes:", error);

}

}

function determinarNivelAlerta(historico, futura) {

const maiorRisco = Math.max(historico, futura);

if (maiorRisco >= 30) { return "#FF0000"; }

else if (maiorRisco >= 20) { return "#FFA500"; }

else if (maiorRisco >= 10) { return "#FFFF00"; }

else { return "#008000"; }

}

async function fetchAndRenderPluvioChart(latitude, longitude, nomeLocal, periodoHoras) {

dashboardTitle.textContent = `Carregando dados de ${periodoHoras}h para ${nomeLocal}...`;

const API\_PLUVIO\_URL = `/api/historical\_pluvio?lat=${latitude}&lon=${longitude}&periodo=${periodoHoras}&nome\_cidade=${encodeURIComponent(nomeLocal)}`;

try {

const response = await fetch(API\_PLUVIO\_URL);

const data = await response.json();

if (!response.ok || data.error || data.volume\_pluviometria === undefined) {

throw new Error(`Erro: ${data.error || 'Dados incompletos da API.'}`);

}

const hourlyData = data.volume\_pluviometria || [];

const labels = data.data\_pluviometria.map(t => {

const date = new Date(t + 'Z');

return date.toLocaleTimeString('pt-BR', { day: '2-digit', month: '2-digit', hour: '2-digit', minute: '2-digit' }).replace(' ', '\n');

});

let acumulado = 0;

const acumuladoArray = hourlyData.map(val => {

if (val !== null) { acumulado += val; }

return acumulado;

});

const pluvioCtx = document.getElementById('pluvioChart').getContext('2d');

if (currentCharts.pluvio) currentCharts.pluvio.destroy();

currentCharts.pluvio = new Chart(pluvioCtx, {

type: 'bar',

data: {

labels: labels,

datasets: [

{ label: 'Pluviometria (mm)', data: hourlyData, backgroundColor: 'rgba(54, 162, 235, 0.7)', yAxisID: 'y-pluv', order: 2 },

{ label: `Acumulado ${periodoHoras}h: ${data.acumulado\_total.toFixed(1)} mm`, data: acumuladoArray, type: 'line', borderColor: 'black', borderWidth: 2, pointRadius: 0, fill: false, yAxisID: 'y-pluv', order: 1 }

]

},

options: {

responsive: true, maintainAspectRatio: false,

scales: {

y: { id: 'y-pluv', type: 'linear', position: 'left', beginAtZero: true, title: { display: true, text: 'Pluviometria (mm)' } },

x: { ticks: { autoSkip: true, maxRotation: 45, minRotation: 45, font: { size: 10 } } }

}

}

});

dashboardTitle.textContent = `Pluviometria Histórica para ${data.cidade}`;

municipioDisplay.textContent = `Município: ${data.municipio} | Total Acumulado: ${data.acumulado\_total.toFixed(1)} mm`;

} catch (error) {

console.error("Falha ao carregar Gráfico de Pluviometria:", error);

dashboardTitle.textContent = `ERRO ao carregar Pluviometria: ${error.message}`;

municipioDisplay.textContent = 'Verifique as datas históricas (máximo 4 dias atrás).';

}

}

async function fetchAndRenderForecastChart(latitude, longitude, nomeLocal) {

const API\_CHART\_URL = `/api/forecast\_chart?lat=${latitude}&lon=${longitude}&nome\_cidade=${encodeURIComponent(nomeLocal)}`;

try {

const forecastResponse = await fetch(API\_CHART\_URL);

const forecastData = await forecastResponse.json();

if (!forecastResponse.ok || forecastData.error || !forecastData.hourly) {

throw new Error(`Erro: ${forecastData.error || 'Dados de previsão incompletos.'}`);

}

const hourlyData = forecastData.hourly.precipitation.slice(0, 72) || [];

const timeLabels = forecastData.hourly.time.slice(0, 72) || [];

let acumulado = 0;

const acumuladoArray = hourlyData.map(val => {

if (val !== null) { acumulado += val; }

return acumulado;

});

const acumuladoTotal = acumulado;

const labels = timeLabels.map(t => {

const date = new Date(t);

return date.toLocaleTimeString('pt-BR', { day: '2-digit', month: '2-digit', hour: '2-digit', minute: '2-digit' }).replace(' ', '\n');

});

const previsaoCtx = document.getElementById('previsaoChart').getContext('2d');

if (currentCharts.previsao) currentCharts.previsao.destroy();

currentCharts.previsao = new Chart(previsaoCtx, {

type: 'bar',

data: {

labels: labels,

datasets: [

{ label: 'Pluviometria Prevista (mm)', data: hourlyData, backgroundColor: 'rgba(255, 159, 64, 0.7)', yAxisID: 'y-pluv', order: 2 },

{ label: `Acumulado Futuro: ${acumuladoTotal.toFixed(1)} mm`, data: acumuladoArray, type: 'line', borderColor: '#FF4500', borderWidth: 2, pointRadius: 0, fill: false, yAxisID: 'y-pluv', order: 1 }

]

},

options: {

responsive: true, maintainAspectRatio: false,

scales: {

y: { id: 'y-pluv', type: 'linear', position: 'left', beginAtZero: true, title: { display: true, text: 'Pluviometria (mm)' } },

x: { ticks: { autoSkip: true, maxRotation: 45, minRotation: 45, font: { size: 10 } } }

}

}

});

} catch (error) {

console.error("Falha ao carregar Gráfico de Previsão:", error);

}

}

// INICIALIZAÇÃO DO MAPA E FLUXO DE DADOS

const map = L.map('map').setView([INITIAL\_LATITUDE, INITIAL\_LONGITUDE], ZOOM\_LEVEL);

L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {

attribution: '&copy; <a href="http://osm.org/copyright">OpenStreetMap</a> contributors'

}).addTo(map);

const geocoder = L.Control.geocoder({

position: 'topleft',

collapsed: false,

placeholder: 'Pesquisar cidade aqui...',

title: 'Pesquisar uma nova localização',

defaultMarkGeocode: false

}).addTo(map);

geocoder.on('markgeocode', function (e) {

const lat = e.geocode.center.lat;

const lon = e.geocode.center.lng;

const nomeCurto = e.geocode.name.split(',')[0].trim();

map.setView([lat, lon], 13);

processarNovoLocal(lat, lon, nomeCurto);

});

function processarNovoLocal(latitude, longitude, nomeLocal) {

if (updateIntervalId) { clearInterval(updateIntervalId); updateIntervalId = null; }

currentCoords = { lat: latitude, lon: longitude };

currentCityName = nomeLocal;

nomeCidadeElement.textContent = nomeLocal;

carregarDadosClimaticos(latitude, longitude, nomeLocal);

updateIntervalId = setInterval(() => { carregarDadosClimaticos(currentCoords.lat, currentCoords.lon, currentCityName); }, 60000);

}

async function carregarDadosClimaticos(latitude, longitude, nomeLocal) {

const API\_URL = `/api/weather?lat=${latitude}&lon=${longitude}&nome\_cidade=${encodeURIComponent(nomeLocal)}`;

tempElement.textContent = "Carregando...";

chuvaHistElement.textContent = "Carregando...";

chuvaFutElement.textContent = "Carregando...";

nomeCidadeElement.textContent = nomeLocal;

alertaPanel.style.backgroundColor = 'white';

const alertaTituloElement = document.getElementById('alerta-titulo');

try {

const resposta = await fetch(API\_URL);

if (!resposta.ok) {

const erro = await resposta.json();

throw new Error(`Erro ${resposta.status}: ${erro.error}`);

}

const dados = await resposta.json();

const nomeReal = dados.cidade\_nome || nomeLocal;

const volumeChuvaHist = dados.chuva\_72h\_hist || 0;

const volumeChuvaFut = dados.chuva\_72h\_fut || 0;

const corAlerta = determinarNivelAlerta(volumeChuvaHist, volumeChuvaFut);

// 🛑 Lógica para obter o nome do nível

let nivelNome = colorToRiskMap[corAlerta] || 'Nível Indefinido';

// 🛑 ATUALIZA O TÍTULO

alertaTituloElement.textContent = `Nível: ${nivelNome}`;

// 🛑 ATUALIZA O FUNDO E AS CLASSES DE COR DO TEXTO

alertaPanel.style.backgroundColor = corAlerta;

if (corAlerta === "#FFFF00" || corAlerta === "#008000") {

alertaPanel.classList.add('color-dark');

alertaPanel.classList.remove('color-light');

} else {

alertaPanel.classList.add('color-light');

alertaPanel.classList.remove('color-dark');

}

nomeCidadeElement.textContent = nomeReal;

tempElement.textContent = dados.temperatura.toFixed(2) + "°C";

chuvaHistElement.textContent = volumeChuvaHist.toFixed(1) + " mm";

chuvaFutElement.textContent = volumeChuvaFut.toFixed(1) + " mm";

} catch (error) {

// 🛑 Tratamento de erro para o título

alertaTituloElement.textContent = `ERRO DE DADOS`;

alertaPanel.style.backgroundColor = 'white';

alertaPanel.classList.add('color-dark');

alertaPanel.classList.remove('color-light');

nomeCidadeElement.textContent = "ERRO NA BUSCA";

tempElement.textContent = "ERRO"; chuvaHistElement.textContent = "ERRO"; chuvaFutElement.textContent = "ERRO";

}

}

async function fetchAndPlaceCapitaisMarkers() {

try {

const response = await fetch('/api/capitais\_risco');

const capitais = await response.json();

if (!response.ok) throw new Error('Falha ao buscar capitais');

capitais.forEach(c => {

if (c.risco\_nivel === 'ERRO') return;

const icon = L.divIcon({

className: `capital-marker-icon nivel-${c.risco\_nivel}`,

html: `<b>${Math.round(c.maior\_risco\_valor || 0)}</b>`,

iconSize: [30, 30],

iconAnchor: [15, 15]

});

const marker = L.marker([c.lat, c.lon], { icon }).addTo(map)

.bindTooltip(`${c.capital} (${c.estado})`)

.on('click', () => {

map.setView([c.lat, c.lon], 11);

processarNovoLocal(c.lat, c.lon, c.capital);

});

});

} catch (error) {

console.error("Erro nos marcadores das capitais:", error);

}

}

async function fetchAndPlaceRiscoMarkers() {

try {

const response = await fetch('/api/cidades\_risco');

const cidadesDeRisco = await response.json();

cidadesDeRisco.forEach(cidade => {

const marker = L.marker([cidade.lat, cidade.lon]).addTo(map)

.bindTooltip(cidade.nome)

.on('click', () => {

map.setView([cidade.lat, cidade.lon], 11);

processarNovoLocal(cidade.lat, cidade.lon, cidade.nome);

});

});

} catch (error) {

console.error("Erro ao colocar marcadores de risco:", error);

}

}

// INICIALIZAÇÃO

processarNovoLocal(INITIAL\_LATITUDE, INITIAL\_LONGITUDE, "Brasília");

fetchAndPlaceCapitaisMarkers();

fetchAndPlaceRiscoMarkers();

Style - /\* ========================================================== \*/

/\* ESTILOS PARA DISPOSITIVOS MÓVEIS (RESPONSIVIDADE) \*/

/\* ========================================================== \*/

@media (max-width: 768px) {

/\* Ajusta o cabeçalho para telas pequenas \*/

.app-header {

height: auto; /\* Altura automática para caber o conteúdo \*/

padding: 15px;

}

.header-top-row {

flex-direction: column; /\* Empilha os itens verticalmente \*/

align-items: flex-start; /\* Alinha à esquerda \*/

gap: 15px; /\* Espaço entre o logo e os botões \*/

}

.app-logo {

font-size: 1.5em; /\* Reduz um pouco o logo \*/

}

/\* Reposiciona a barra de pesquisa para ocupar a largura \*/

.leaflet-control-geocoder {

top: 130px !important; /\* Desce para baixo do cabeçalho \*/

left: 10px !important;

right: 10px !important;

width: auto !important; /\* Ocupa a largura disponível \*/

}

/\* Reposiciona o painel de dados para a parte de baixo \*/

.data-panel {

top: auto; /\* Remove o 'top' fixo \*/

bottom: 20px; /\* Posiciona na base \*/

left: 10px;

right: 10px;

width: auto; /\* Largura automática \*/

z-index: 1001; /\* Garante que fique acima de outros elementos do mapa \*/

}

/\* Ajusta o dashboard para telas pequenas \*/

.charts-container {

flex-direction: column; /\* Gráficos sempre um em cima do outro \*/

}

#pluviometria-card, #previsao-card {

min-width: 90%; /\* Remove a largura mínima fixa \*/

height: 40vh; /\* Altura um pouco menor \*/

}

.seletor-container, .dashboard-header {

flex-direction: column;

align-items: stretch; /\* Itens ocupam toda a largura \*/

text-align: center;

}

#back-to-map-btn {

position: static; /\* Remove o posicionamento absoluto \*/

transform: none;

margin-top: 10px;

}

/\* Ajusta a grade de cidades na Sala de Monitoramento \*/

#capitais-grid-container {

/\* A configuração atual com auto-fill já é bem responsiva! \*/

/\* Podemos apenas garantir que o mínimo seja um pouco menor se necessário \*/

grid-template-columns: repeat(auto-fill, minmax(280px, 1fr));

}

}

Welcome - <!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="UTF-8">

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

<title>RiskGeo Solutions - Monitoramento Climático</title>

<script src="https://cdn.jsdelivr.net/npm/chart.js@4.4.0/dist/chart.umd.min.js"></script>

<link rel="stylesheet" href="https://unpkg.com/leaflet/dist/leaflet.css" />

<link rel="stylesheet" href="https://unpkg.com/leaflet-control-geocoder/dist/Control.Geocoder.css" />

<link rel="stylesheet" href="style.css">

<style>

:root {

--cor-faixa-escura: rgba(0, 50, 150, 0.85);

--cor-texto-claro: white;

--cor-texto-escuro: #003296;

--cor-fundo-dashboard: #F0F8FF;

}

body { margin: 0; padding: 0; font-family: 'Arial', sans-serif; }

#map-container, #dashboard-container, #monitor-room-container { position: absolute; top: 0; left: 0; width: 100%; height: 100vh; transition: opacity 0.5s; }

#map-container { z-index: 100; }

#map { width: 100%; height: 100%; }

.app-header { position: absolute; top: 0; left: 0; right: 0; height: 75px; background-color: var(--cor-faixa-escura); color: var(--cor-texto-claro); z-index: 1001; padding: 0 25px; display: flex; flex-direction: column; justify-content: center; align-items: flex-start; }

.header-top-row { width: 100%; display: flex; justify-content: space-between; align-items: center; padding-bottom: 5px; }

.app-logo { font-size: 1.8em; font-weight: 700; letter-spacing: 1px; color: var(--cor-texto-claro); text-decoration: none; cursor: pointer; }

.aviso-legal { font-size: 0.75em; color: #FFD700; width: 100%; text-align: left; margin-top: -5px; }

.header-links-group { display: flex; align-items: center; gap: 15px; }

.header-link, .header-link:visited { color: var(--cor-texto-claro); text-decoration: none; font-size: 1.1em; padding: 7px 12px; border: 1px solid var(--cor-texto-claro); border-radius: 4px; transition: background-color 0.3s; }

.header-link:hover { background-color: rgba(255, 255, 255, 0.2); }

.leaflet-control-geocoder { position: absolute; top: 85px !important; left: 10px !important; z-index: 999; width: 245px !important; max-width: 300px !important; }

.leaflet-control-geocoder input { width: 100% !important; font-size: 22px !important; padding-top: 8px !important; padding-bottom: 8px !important; background-color: #000000 !important; color: white; border: 1px solid white; }

.leaflet-control-geocoder input::placeholder { color: #ccc; opacity: 1; }

#welcome-container { width: 100%; height: 100vh; display: flex; justify-content: center; align-items: center; color: white; text-align: center; position: relative; overflow: hidden; }

#bg-video { position: absolute; top: 50%; left: 50%; min-width: 100%; min-height: 100%; width: auto; height: auto; z-index: -100; transform: translateX(-50%) translateY(-50%); background-size: cover; }

.welcome-overlay { background: rgba(0, 20, 60, 0.6); padding: 40px; border-radius: 15px; max-width: 800px; }

.welcome-logo-container { display: flex; align-items: center; justify-content: center; gap: 20px; margin-bottom: 20px; }

.welcome-logo { width: 100px; height: 100px; }

.welcome-title { font-size: 3.5em; font-weight: bold; text-shadow: 2px 2px 4px #000000; }

.welcome-subtitle { font-size: 1.5em; margin-bottom: 30px; }

.welcome-features { list-style: none; padding: 0; margin-bottom: 40px; text-align: left; display: inline-block; }

.welcome-features li { font-size: 1.2em; margin-bottom: 15px; display: flex; align-items: center; }

.welcome-features li::before { content: '✓'; color: #00FF00; font-size: 1.5em; margin-right: 15px; }

.enter-btn { background-color: #28a745; color: white; padding: 15px 40px; text-decoration: none; font-size: 1.5em; font-weight: bold; border-radius: 8px; transition: background-color 0.3s, transform 0.2s; }

.enter-btn:hover { background-color: #218838; transform: scale(1.05); }

</style>

</head>

<body>

<div id="welcome-container">

<video autoplay muted loop id="bg-video">

<source src="terra.mp4" type="video/mp4">

Seu navegador não suporta vídeos.

</video>

<div class="welcome-overlay">

<div class="welcome-logo-container">

<img src="logo-riskgeo360.png" alt="Logo RiskGeo 360" class="welcome-logo">

<h1 class="welcome-title">RISKGEO 360</h1>

</div>

<p class="welcome-subtitle">MONITORAMENTO GEOCLIMÁTICO INTELIGENTE.</p>

<ul class="welcome-features">

<li>VISUALIZAÇÃO DE DADOS CLIMÁTICOS EM MAPA INTERATIVO.</li>

<li>BUSCA DE INFORMAÇÕES PARA QUALQUER LOCALIDADE.</li>

<li>MONITORAMENTO EM TEMPO REAL DAS PRINCIPAIS CIDADES.</li>

<li>ANÁLISE DETALHADA COM GRÁFICOS DE PLUVIOMETRIA.</li>

</ul>

<a href="/index.html" class="enter-btn">ENTRAR</a>

</div>

</div>

</body>

</html>