Projeto de Monitoramento da Qualidade da Água das Praias de Praia Grande

1. Descrição do Site a Ser Monitorado

URL:
https://arcgis.cetesb.sp.gov.br/portal/apps/experiencebuilder/experience/?id=bdd0cbd4bf094df9a000bf663254c21f&page=Classifica%C3%A7%C3%A3o-Atual

O site monitorado é uma aplicação interativa desenvolvida pela CETESB (Companhia Ambiental do Estado de São Paulo) utilizando a plataforma ArcGIS Experience Builder. Seu propósito é disponibilizar ao público a classificação da qualidade da água de balneabilidade das praias do litoral paulista, indicando se estão próprias ou impróprias para banho. A interface principal é um mapa interativo onde os usuários podem visualizar a situação de cada praia, geralmente por meio de marcadores coloridos e informações detalhadas em pop-ups ao clicar neles.

É importante notar que, por ser uma aplicação dinâmica baseada em ArcGIS, os dados não estão em tabelas HTML estáticas. Em vez disso, são carregados via chamadas a uma API REST (Application Programming Interface) subjacente, que fornece os dados em formato JSON (JavaScript Object Notation).

2. Proposta de Monitoramento

A proposta deste projeto é automatizar a coleta dos dados de balneabilidade das praias da cidade de Praia Grande, localizada na Baixada Santista, diretamente do portal da CETESB. O objetivo é obter de forma programática a classificação atualizada da qualidade da água para cada praia do município, permitindo uma análise rápida e um acompanhamento contínuo sem a necessidade de interação manual com o mapa.

Os dados coletados incluirão informações como o nome da praia, sua classificação de balneabilidade (própria/imprópria), e as datas de coleta e atualização das informações. Essa base de dados será essencial para futuras análises e para embasar decisões relacionadas ao uso recreativo dessas praias.

3. Descrição Passo a Passo de Como o Monitoramento Foi Realizado

O monitoramento foi realizado utilizando a linguagem de programação Python, aproveitando bibliotecas especializadas em automação de navegador (selenium) e requisições HTTP (requests), além de manipulação de dados (pandas). A estratégia principal foi identificar a API subjacente que alimenta o mapa com os dados de qualidade da água e, então, fazer uma requisição direta a essa API para obter os dados em formato JSON, que é mais fácil de processar.

Passo 3.1: Instalação e Importação das Bibliotecas Necessárias

Este passo garante que todas as ferramentas que o script Python precisa estejam disponíveis.

Comandos Executados (no terminal ou prompt de comando, uma única vez para o ambiente):

Bash
conda install -c conda-forge selenium beautifulsoup4 pandas requests # Instalação via conda
pip install webdriver-manager # Instalação via pip (recomendado para gerenciar o driver do navegador)

Comando Executado (na Célula 1 do Jupyter Notebook):

Python

```
# Célula 1: Importar Bibliotecas
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
import time
import json
import pandas as pd
import requests
from bs4 import BeautifulSoup

print("Bibliotecas importadas com sucesso!")
```

Resultados Obtidos:

```
Bibliotecas importadas com sucesso!
```

Passo 3.2: Configuração do Navegador e Acesso ao Site da CETESB

Neste passo, inicializamos o navegador Chrome (controlado pelo Selenium) e o direcionamos para a URL do portal da CETESB. Uma espera de tempo é adicionada para garantir que todo o conteúdo dinâmico do mapa seja carregado.

Comando Executado (na Célula 2 do Jupyter Notebook):

```
Python
# Célula 2: Configurar WebDriver e Navegar para o Site

service = Service()
options = webdriver.ChromeOptions()

# Para depuração, o modo headless estava desativado. Para uso final, pode ser reativado.
```

```
# options.add_argument('--headless')
options.add_argument('--no-sandbox')
options.add_argument('--disable-dev-shm-usage')
options.add_argument('--start-maximized')

driver = webdriver.Chrome(service=service, options=options)

url =
'https://arcgis.cetesb.sp.gov.br/portal/apps/experiencebuilder/experience/?id=bdd0cbd4bf094df9a000bf663254c21f&page=Classifica%C3%A7%C3%A3o-Atual'
driver.get(url)
print(f"Navegado para: {url}")

time.sleep(15) # Espera para carregar a aplicação do mapa
print("Carregamento inicial da página concluído. A aplicação do mapa deve estar visível.")
```

Resultados Obtidos:

Navegado para:
https://arcgis.cetesb.sp.gov.br/portal/apps/experiencebuilder/experience/?id=bdd0cbd4bf094df9a000bf663254c21f&page=Classifica%C3%A7%C3%A3o-Atual
Carregamento inicial da página concluído. A aplicação do mapa deve estar visível.

(Neste ponto, uma janela do navegador Chrome foi aberta e carregou o mapa da CETESB.)

Passo 3.3: Identificação da API de Dados (Processo Manual de Depuração)

Este foi um passo crucial de investigação. Como os dados eram carregados dinamicamente por JavaScript, não era possível simplesmente "clicar" em um elemento HTML ou extrair de uma tabela estática. A solução foi identificar a requisição de rede que a própria aplicação do mapa fazia para obter os dados.

Processo Manual Detalhado:

1. Com o navegador aberto pelo Selenium (após a Célula 2), as Ferramentas do Desenvolvedor (F12) foram abertas na aba **"Rede" (Network)**.
2. O filtro para requisições **"XHR"** ou **"Fetch"** foi aplicado.
3. Ao interagir com o mapa (zoom, pan, ou clique em um marcador de praia), uma requisição específica foi observada e identificada como a fonte dos dados de qualidade da água. A URL dessa requisição foi cuidadosamente inspecionada.
4. Foi notado que a requisição original usava o formato f=pbf (Protocol Buffer, um formato binário otimizado para o mapa). Para facilitar a leitura e o processamento em Python, o parâmetro f precisou ser alterado para f=json.
5. A estrutura do JSON retornado pela API foi examinada na aba "Pré-visualização" (Preview) para identificar os nomes exatos dos campos de interesse, como municipio, praia, classificacao_texto, data_amostra_inicio, data_atual, etc. Esta etapa foi fundamental para corrigir os erros de "Field name does not exist" encontrados nas tentativas iniciais.

Comando Executado (Célula 3 do Jupyter Notebook):

Python

```
# Célula 3: Identificar a Fonte de Dados via Aba Rede (PASSO MANUAL - CRÍTICO!)

print("--- PASSOS MANUAIS DE DEBUGGING ---")
print("1. Após executar a Célula 2, a janela do navegador com o mapa da CETESB deve estar aberta.")
print("2. Abra as Ferramentas do Desenvolvedor (pressione F12) nessa janela do navegador.")
print("3. Vá para a aba 'Rede' (ou 'Network').")
print("4. Filtre por requisições 'XHR' ou 'Fetch' (isso mostra as chamadas de API).")
print("5. No mapa, clique manualmente em um marcador de praia em Praia Grande ou aproxime o zoom até que os dados das praias de Praia Grande apareçam/atualizem.")
print("6. Observe a aba 'Rede'. Procure por requisições que começam com 'query', 'FeatureServer', 'MapServer' ou que pareçam estar buscando dados de pontos no mapa.")
print("   Especialmente, procure por requisições que retornem dados JSON contendo nomes de praias, status de qualidade da água, etc.")
print("7. Clique com o botão direito em uma dessas requisições relevantes.")
print("8. Selecione 'Copiar' -> 'Copiar como cURL (bash)' ou 'Copiar como Fetch'.")
print("9. Alternativamente, examine as abas 'Cabeçalhos' (Headers) e 'Pré-visualização' (Preview) / 'Resposta' (Response) da requisição para entender a URL e os parâmetros e os nomes dos campos.")
```

print("\nUma vez que você identificou o endpoint da API e os parâmetros, você pode prosseguir para a Célula 4.")

Resultados Obtidos (Observações Manuais e Correções):

- **URL da API identificada:** https://arcgis.cetesb.sp.gov.br/server/rest/services/Hosted/Praias/FeatureServer/0/query
- **Parâmetros-chave observados e corrigidos:**
  - f: Alterado de pbf para json.
  - where: Definido como "municipio = 'PRAIA GRANDE'" (campo municipio em minúsculas).
  - outFields: Definido como * para obter todos os campos.
- **Nomes exatos dos campos no JSON:** municipio, praia, classificacao_texto, data_amostra_inicio, data_atual, ugrhi, dist_norte, qualidade, id_praia, id_municipio, cod_mun_ibge, objectid.

Passo 3.4: Coleta de Dados via Requisição Direta à API e Processamento

Com a URL da API e os nomes dos campos corretos em mãos, uma requisição HTTP direta foi feita usando a biblioteca requests. A resposta JSON foi então parseada e convertida em um DataFrame do pandas, que é uma estrutura de dados ideal para análise. As datas, que vieram como timestamps (milissegundos), foram convertidas para o formato datetime legível.

Comando Executado (Célula 4 do Jupyter Notebook):

Python
Célula 4: Obter Dados da API (FINALMENTE CORRIGIDA!)

api_url_base = "https://arcgis.cetesb.sp.gov.br/server/rest/services/Hosted/Praias/FeatureServer/0/query"

```
# Parâmetros ajustados com o nome exato do campo 'municipio'
params = {
  'f': 'json', # Formato de resposta JSON
  'where': "municipio = 'PRAIA GRANDE'", # Campo 'municipio' em minúsculas, como no JSON
  'returnGeometry': 'false',
  'spatialRel': 'esriSpatialRelIntersects',
  'outFields': '*', # Mantém para obter todos os campos (boa prática)
  'outSR': '4326',
}

print(f"Tentando buscar dados diretamente da API: {api_url_base}")
print(f"Com os parâmetros: {params}")

try:
  response = requests.get(api_url_base, params=params)
  response.raise_for_status() # Levanta uma exceção para erros HTTP
  data = response.json()

  if 'features' in data:
    beach_data = []
    for feature in data['features']:
      attributes = feature.get('attributes', {})

      # Extração de atributos com os nomes EXATOS encontrados no JSON
      # e conversão de timestamp para data legível
      from datetime import datetime
      data_amostra_inicio_ts = attributes.get('data_amostra_inicio')
      data_atual_ts = attributes.get('data_atual')

      data_amostra_inicio_dt = datetime.fromtimestamp(data_amostra_inicio_ts / 1000) if data_amostra_inicio_ts else 'N/A'
      data_atual_dt = datetime.fromtimestamp(data_atual_ts / 1000) if data_atual_ts else 'N/A'


      beach_info = {
```

```
            'ID_Objeto': attributes.get('objectid', 'N/A'),
            'ID_Praia': attributes.get('id_praia', 'N/A'),
            'ID_Municipio': attributes.get('id_municipio', 'N/A'),
            'Cod_Municipio_IBGE': attributes.get('cod_mun_ibge', 'N/A'),
            'Municipio': attributes.get('municipio', 'N/A'),
            'Praia': attributes.get('praia', 'N/A'),
            'Classificacao_Agua': attributes.get('classificacao_texto', 'N/A'),
            'Data_Amostra_Inicio': data_amostra_inicio_dt,
            'Data_Atualizacao_Sistema': data_atual_dt,
            'UGRHI': attributes.get('ugrhi', 'N/A'),
            'Dist_Norte': attributes.get('dist_norte', 'N/A'),
            'Qualidade_Codigo': attributes.get('qualidade', 'N/A'),
        }
        beach_data.append(beach_info)

    if beach_data:
        df = pd.DataFrame(beach_data)
        print("\nDados Extraídos (primeiras 5 linhas do DataFrame):")
        print(df.head())

        output_filename = 'qualidade_agua_praia_grande.csv'
        df.to_csv(output_filename, index=False, encoding='utf-8')
        print(f"\nDados salvos com sucesso em '{output_filename}'")
    else:
        print("Nenhum dado de praia encontrado para 'Praia Grande'.")

  else:
    print("Nenhuma 'features' (dados de atributos) encontrada na resposta da API.")
    print("Resposta Completa da API:", json.dumps(data, indent=2))

except requests.exceptions.RequestException as e:
  print(f"Erro ao buscar dados da API: {e}")
  if 'response' in locals() and response.text:
    print("Conteúdo da Resposta HTTP:", response.text[:500])
except json.JSONDecodeError:
  print("Erro ao decodificar a resposta JSON da API.")
  if 'response' in locals() and response.text:
    print("Conteúdo da Resposta da API:", response.text[:500])
```

```
except Exception as e:
    print(f"Ocorreu um erro inesperado durante o processamento dos dados da API: {e}")
finally:
    # O driver do Selenium é fechado após a operação principal da API
    driver.quit()
    print("Navegador fechado (se o driver foi fechado).")
```

Resultados Obtidos:

Tentando buscar dados diretamente da API:
https://arcgis.cetesb.sp.gov.br/server/rest/services/Hosted/Praias/FeatureServer/0/query
Com os parâmetros: {'f': 'json', 'where': "municipio = 'PRAIA GRANDE'", 'returnGeometry': 'false', 'spatialRel': 'esriSpatialRelIntersects', 'outFields': '*', 'outSR': '4326'}

Dados Extraídos (primeiras 5 linhas do DataFrame):

	ID_Objeto	ID_Praia	ID_Municipio	Cod_Municipio_IBGE	Municipio
0	134	PGCF390	558	3541000	PRAIA GRANDE
1	135	PGBO393	558	3541000	PRAIA GRANDE
2	138	PGJM398	558	3541000	PRAIA GRANDE
3	144	PGFL411	558	3541000	PRAIA GRANDE
4	137	PGAV397	558	3541000	PRAIA GRANDE

	Praia	Classificacao_Agua	Data_Amostra_Inicio
0	CANTO DO FORTE	Imprópria	2025-05-11
1	BOQUEIRÃO	Imprópria	2025-05-11
2	VILA TUPI	Imprópria	2025-05-11
3	BALNEARIO FLÓRIDA	Imprópria	2025-05-11
4	AVIAÇÃO	Imprópria	2025-05-11

	Data_Atualizacao_Sistema	UGRHI	Dist_Norte	Qualidade_Codigo
0	2025-05-15 12:10:03.363	Baixada Santista	390	0
1	2025-05-15 12:10:03.363	Baixada Santista	393	0
2	2025-05-15 12:10:03.363	Baixada Santista	398	0
3	2025-05-15 12:10:03.363	Baixada Santista	411	0
4	2025-05-15 12:10:03.363	Baixada Santista	397	0

Dados salvos com sucesso em 'qualidade_agua_praia_grande.csv'

Navegador fechado (se o driver foi fechado).

Passo 3.5: Fechar o Navegador

Este passo final garante que a instância do navegador aberta pelo Selenium seja encerrada, liberando os recursos do sistema.

Comando Executado (Célula 5 do Jupyter Notebook):

Python
```
# Célula 5: Fechar o Navegador
driver.quit()
print("Navegador fechado.")
```

Resultados Obtidos:

Navegador fechado.

(Esta mensagem pode ter aparecido já no final da Célula 4, dependendo de como você comentou ou não o driver.quit() lá.)

4. Resultado Final Obtido pelo Monitoramento Realizado

Ao final do processo de monitoramento, foi obtido um arquivo CSV nomeado qualidade_agua_praia_grande.csv. Este arquivo contém os dados tabulados da qualidade da água de todas as praias de Praia Grande, conforme disponibilizado pela CETESB, com a seguinte estrutura:

- **ID_Objeto**: Identificador único do registro.
- **ID_Praia**: Código interno da praia.

- **ID_Municipio**: Código interno do município.
- **Cod_Municipio_IBGE**: Código IBGE do município.
- **Municipio**: Nome do município (PRAIA GRANDE).
- **Praia**: Nome específico da praia (ex: CANTO DO FORTE, BOQUEIRÃO, VILA TUPI, etc.).
- **Classificacao_Agua**: Estado de balneabilidade da água (Própria ou Imprópria).
- **Data_Amostra_Inicio**: Data de início da coleta da amostra (convertida para formato legível).
- **Data_Atualizacao_Sistema**: Data de atualização da informação no sistema da CETESB (convertida para formato legível).
- **UGRHI**: Unidade de Gerenciamento de Recursos Hídricos (Baixada Santista).
- **Dist_Norte**: Distância em relação a um ponto de referência norte (provavelmente em metros).
- **Qualidade_Codigo**: Um código numérico para a qualidade da água (0 para Imprópria, 1 para Própria, etc., dependendo da convenção da CETESB).

Este dataset oferece uma visão instantânea e organizada da condição de balneabilidade das praias de Praia Grande, permitindo análises temporais se o monitoramento for repetido periodicamente.

5. Sugestão de Possível Tomada de Decisão com Base nos Resultados

Com base nos resultados apresentados, a principal tomada de decisão é a **orientação sobre a segurança de entrar no mar**.

Exemplo de Análise e Decisão:

- **Análise do Dataset:** Observando as primeiras linhas do DataFrame obtido:
  - CANTO DO FORTE: **Imprópria**
  - BOQUEIRÃO: **Imprópria**
  - VILA TUPI: **Imprópria**
  - BALNEARIO FLÓRIDA: **Imprópria**
  - AVIAÇÃO: **Imprópria**

- **Interpretação:** Para a data de coleta das amostras, todas as praias listadas em Praia Grande que foram monitoradas e aparecem nesta amostra inicial estão classificadas como **"Impróprias"** para banho. O campo Qualidade_Codigo com valor 0 reforça essa classificação.
- **Tomada de Decisão Sugerida:**
    - **Não entrar no mar:** A recomendação direta, com base nesses dados, seria **evitar o contato com a água** nas praias de Praia Grande atualmente classificadas como "Impróprias". Banhistas e frequentadores da praia devem ser alertados sobre a possível presença de agentes poluidores que podem causar doenças.
    - **Verificar datas de coleta:** É crucial observar a Data_Amostra_Inicio e Data_Atualizacao_Sistema. Se a data de amostragem for muito antiga, a situação pode ter mudado. Este monitoramento automatizado permite que as datas sejam verificadas rapidamente.
    - **Monitoramento contínuo:** Para tomadas de decisão mais dinâmicas (dia a dia ou semana a semana), o projeto pode ser agendado para rodar periodicamente (ex: diariamente ou semanalmente), fornecendo dados atualizados que podem ser publicados em um painel ou enviados como alerta. Isso permitiria que cidadãos e autoridades tivessem informações em tempo real para decidir sobre a balneabilidade.

Este projeto não apenas coleta os dados, mas também estabelece a base para uma análise crítica e uma tomada de decisão informada em relação à saúde pública e ao lazer na costa de Praia Grande.