Plano de Monitoração da Corrosão em Estruturas Metálicas

Clara Ferreira e João Vinícius Duarte, Prof Dr° Robson Marinho da Silva

Salvador, BA, junho de 2024

Universidade do Estado da Bahia (UNEB)

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Sumário

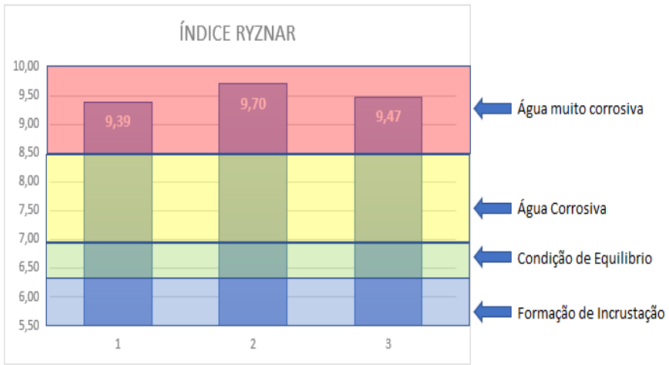
UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Introdução:

A corrosão é um processo natural de deterioração dos metais causado por reações químicas ou eletroquímicas entre o metal e o meio corrosivo. No Brasil, as construções com estruturas metálicas enfrentam desafios devido às variações climáticas, salinidade e à poluição, o que pode causar anomalias nas estruturas, comprometendo sua integridade.

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Introdução:



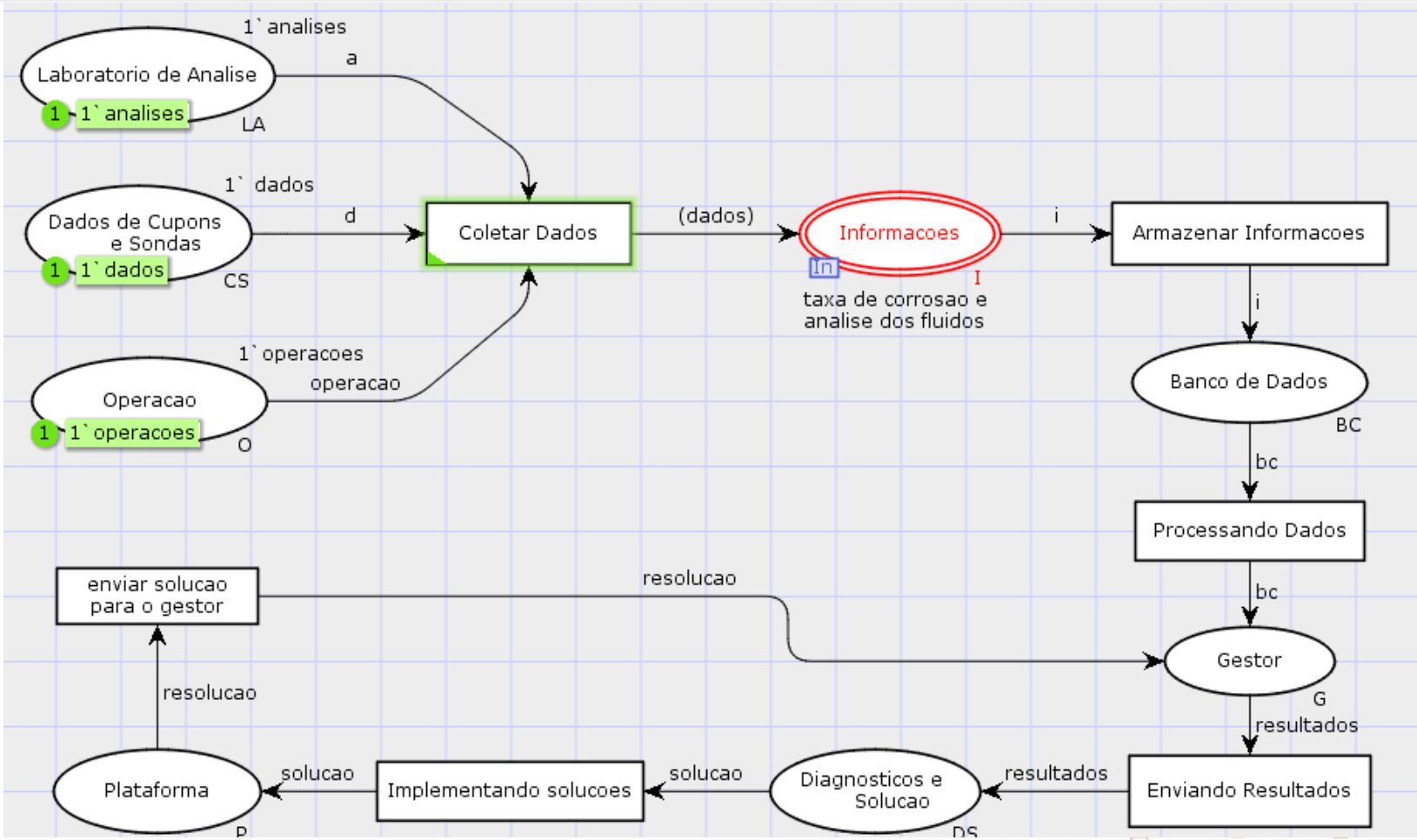
Para avaliar a água em termos de agressividade e incrustação são utilizados métodos de avaliação como o Ph de Saturação, Índice de Saturação proposto por Langelier (ISL) e o índice de estabilidade de Ryznar (IER). Os valores obtidos para a água analisada sob o parâmetro proposto poderá ser analisado através da relação entre o valor obtido para o IER e o estágio da corrosão. A severidade da corrosão, ou seja, a relação do estado de saturação do carbonato de cálcio e a camada formada podem ser classificadas pelo (ISL) e o (IER).

Sumário

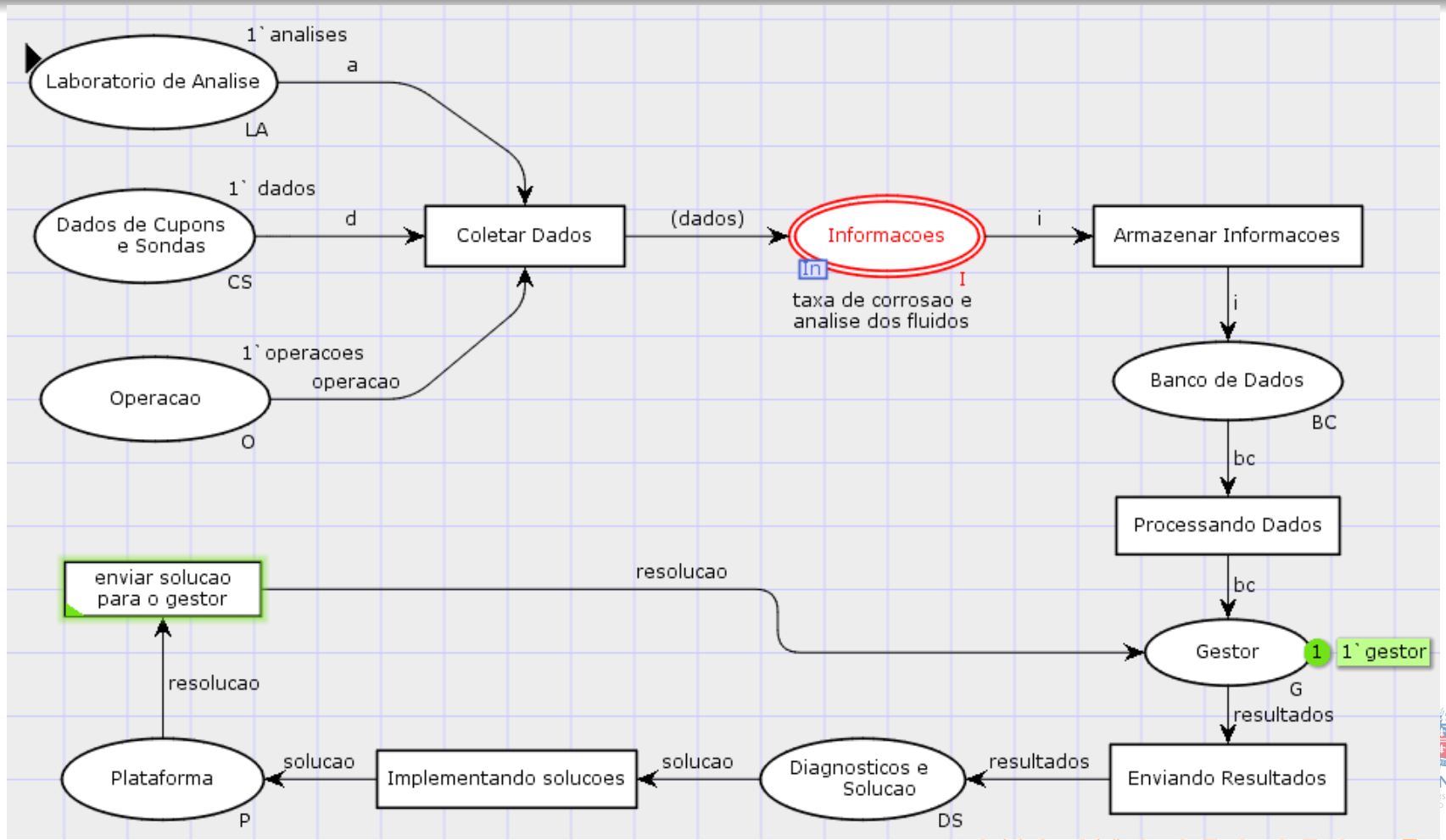
OBJETIVO: A proposta do programa consiste no monitoramento da Taxa de Penetração da Corrosão (TPC), permitindo a sua categorização conforme a intensidade. Adicionalmente, o programa realiza a análise e classificação da qualidade da água, calculando o pH de saturação, o Índice de Saturação de Langelier (ISL) e o Índice de Estabilidade de Ryznar (IER). Os valores obtidos para a água investigada, sob o parâmetro proposto, podem ser analisados através da relação entre o valor do ISL e do IER e o estágio da corrosão. Esses índices são indicadores cruciais para avaliar a propensão da água a causar incrustações ou ser agressiva, informações essenciais para o tratamento adequado e a proteção das estruturas metálicas.

Sumário

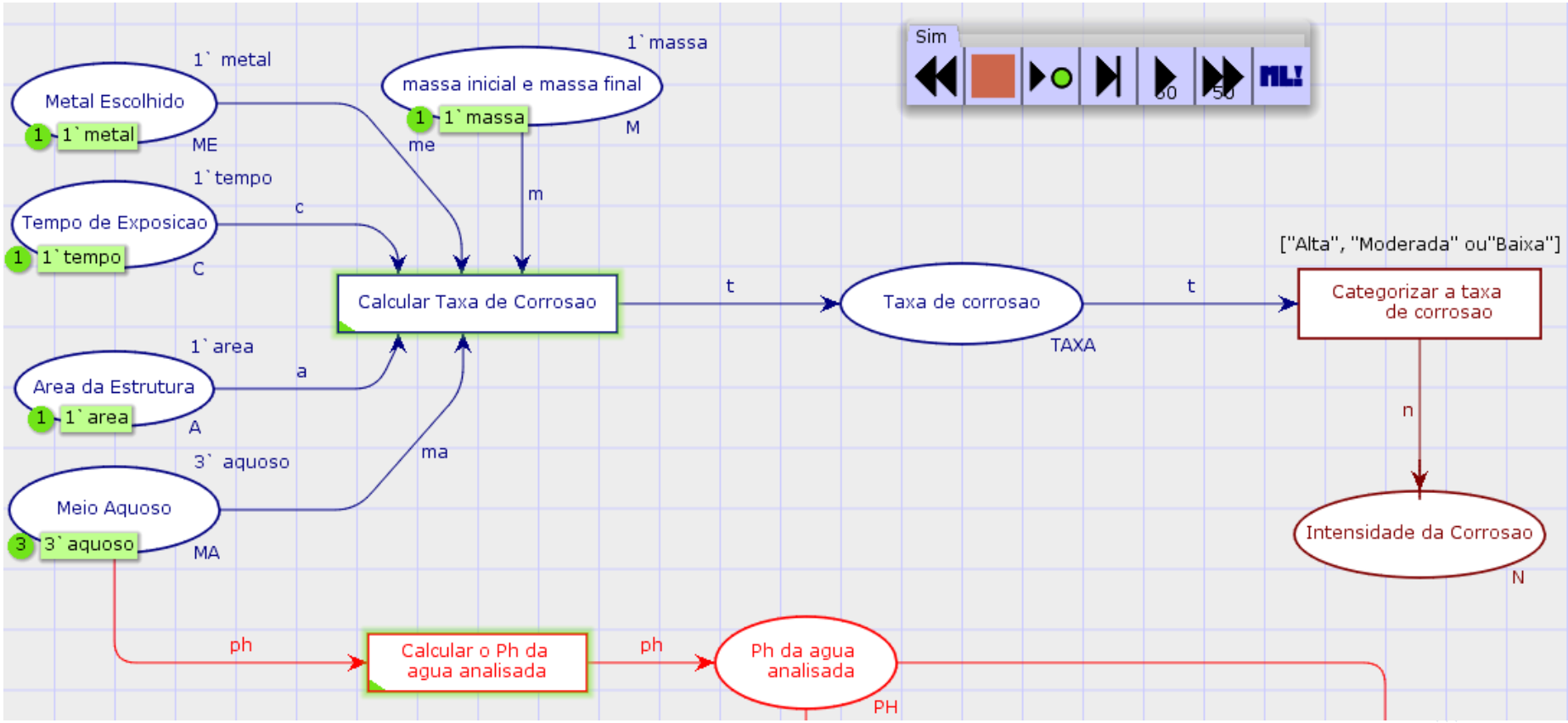
Funcionamento e aplicação do plano de monitoramento



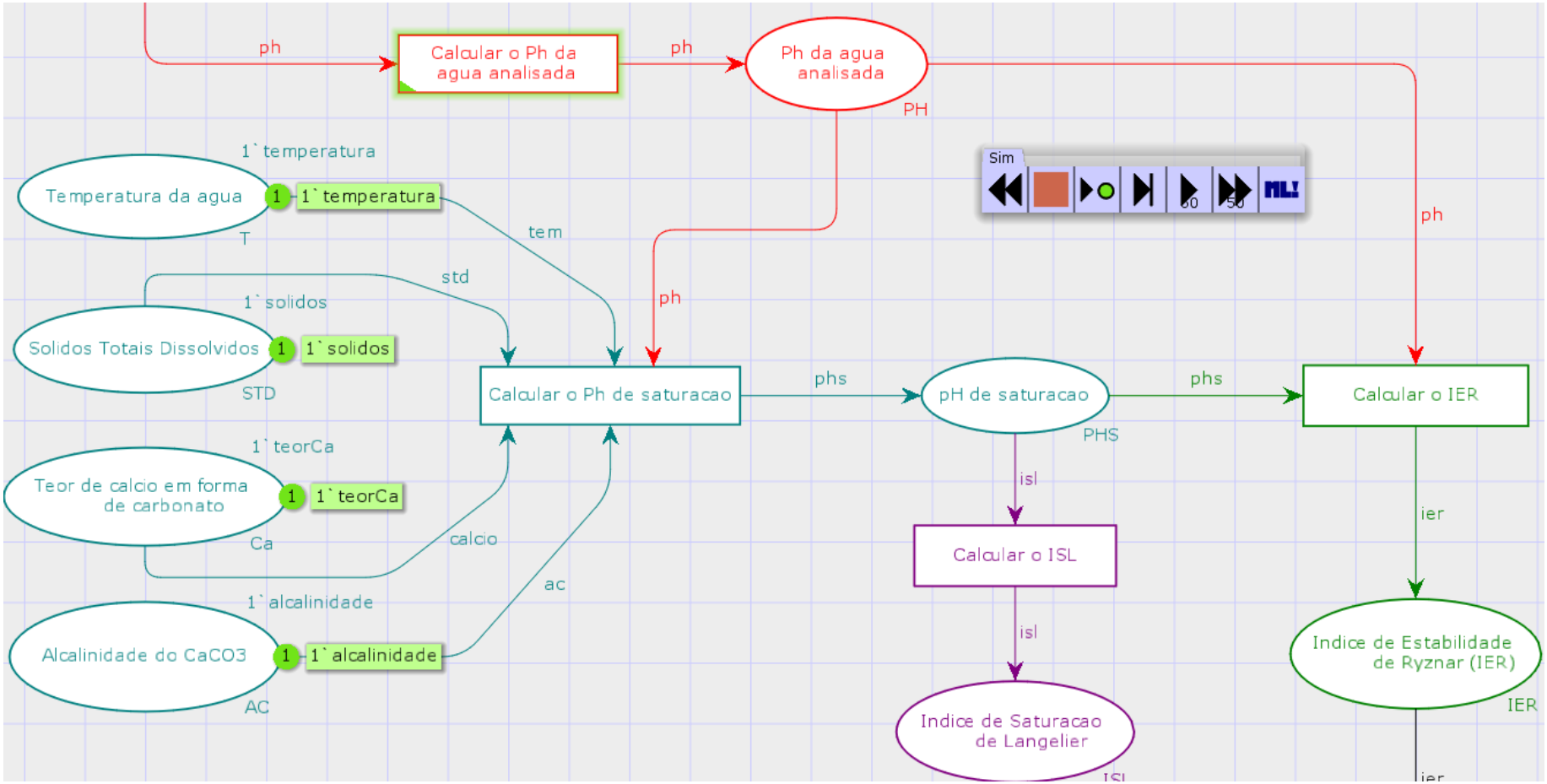
Funcionamento e aplicação do plano de monitoramento



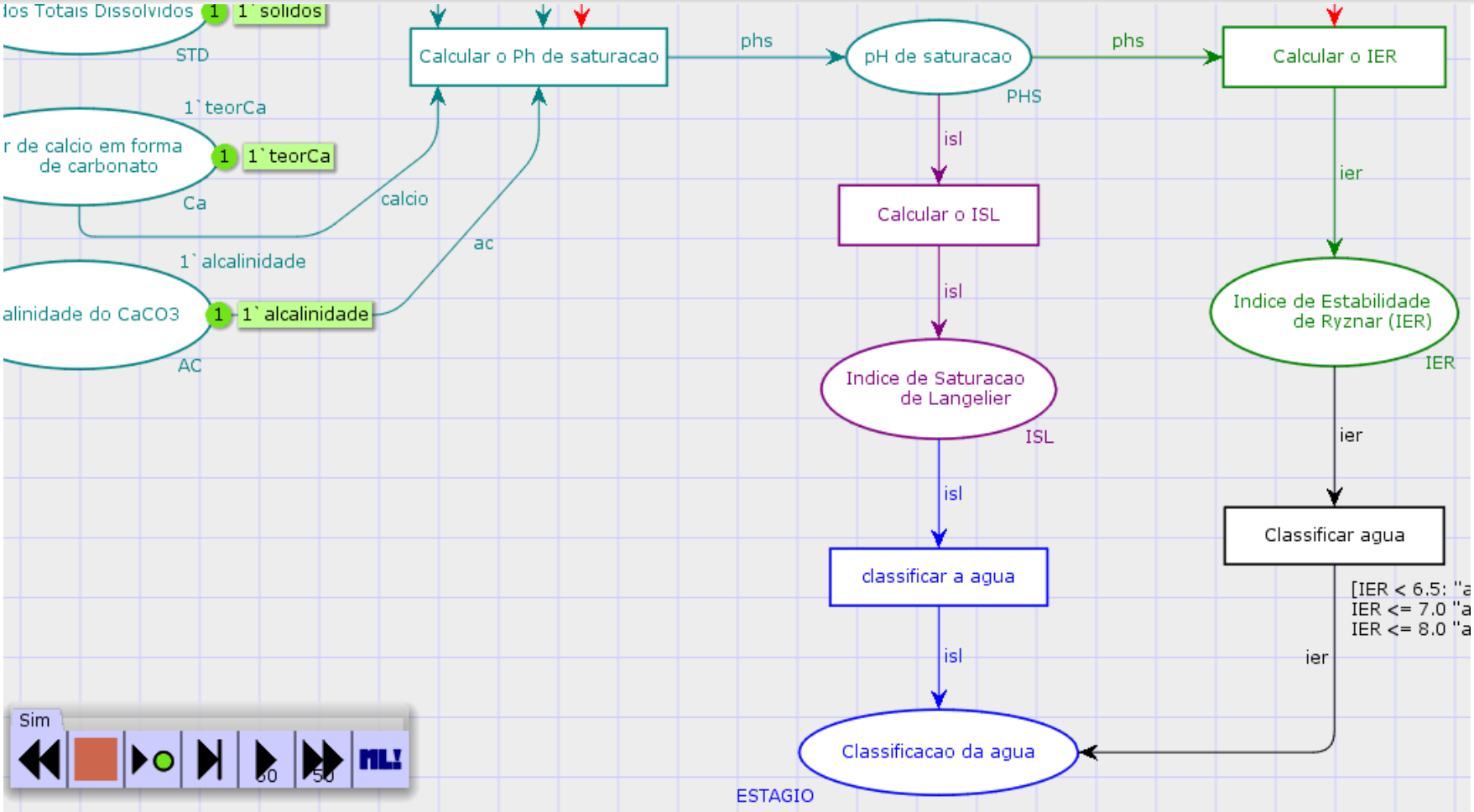
Cálculo da taxa de corrosão e análise da água



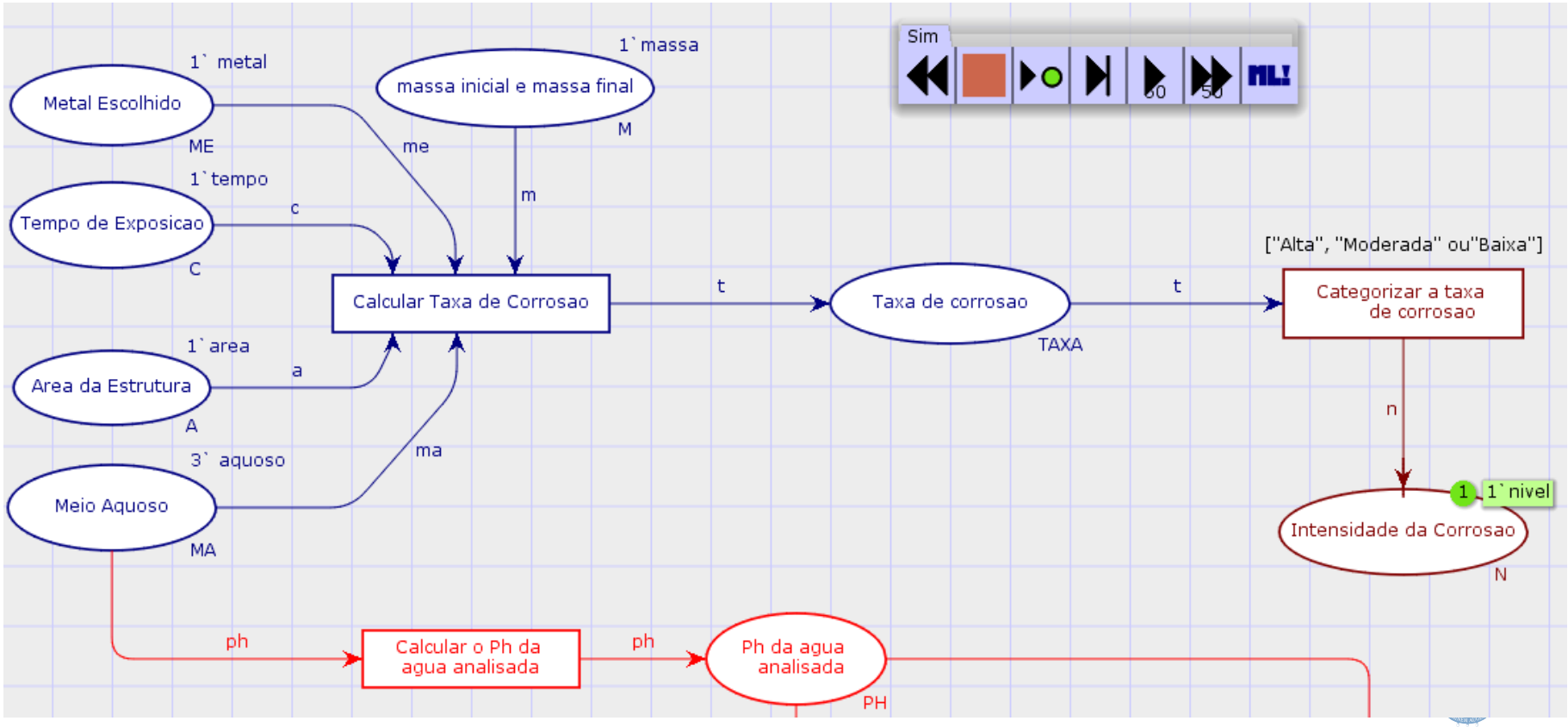
Cálculo da taxa de corrosão e análise da água



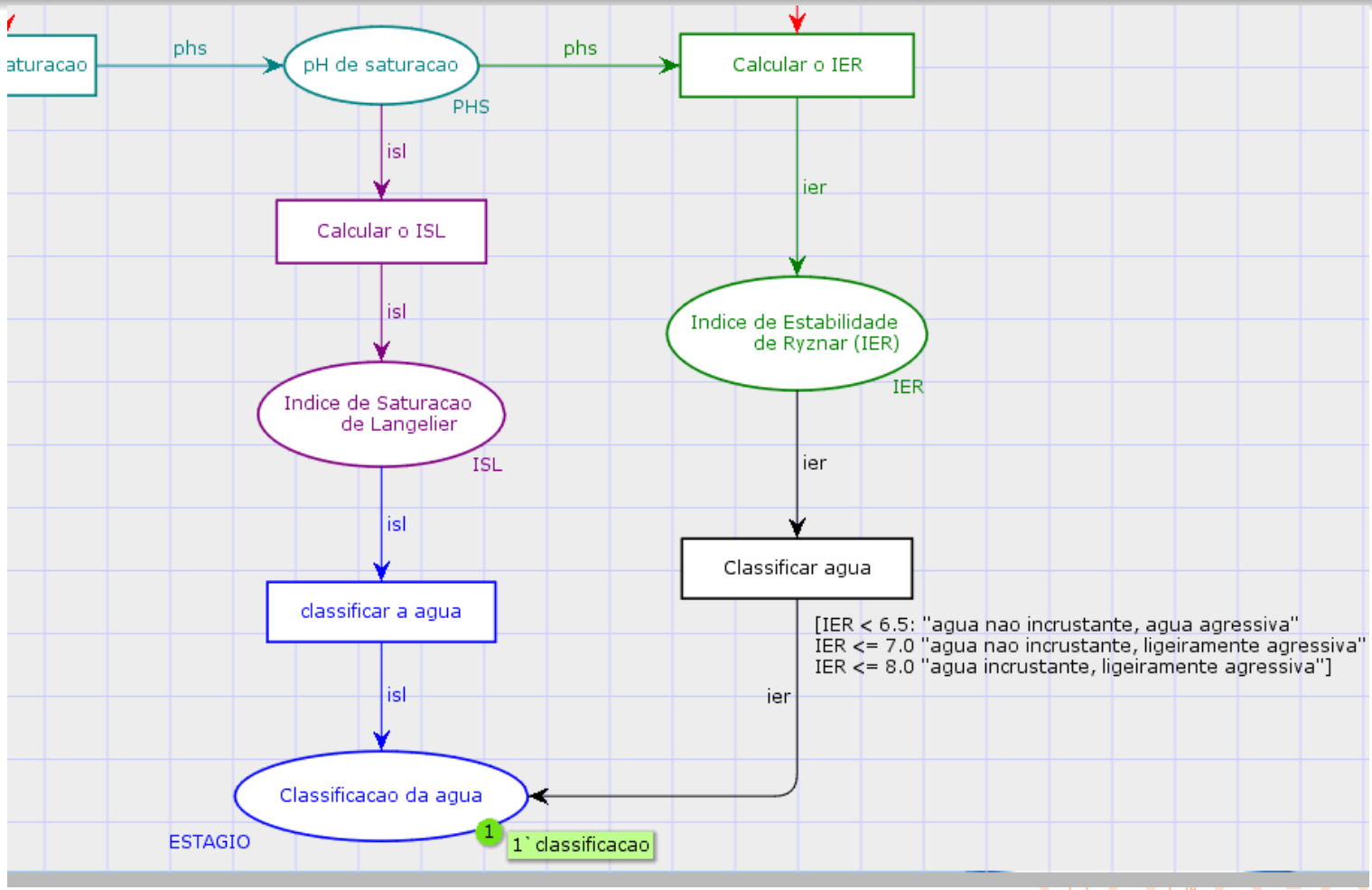
Cálculo da taxa de corrosão e análise da água



Cálculo da taxa de corrosão e análise da água



Cálculo da taxa de corrosão e análise da água



Sumário

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Bibliotecas

- **Tkinter**: é a biblioteca padrão do Python para criar interfaces gráficas de usuário (GUI) e intuitiva para inserir os dados necessários para o monitoramento de corrosão.
- **Pandas**: é uma biblioteca poderosa para manipulação e análise de dados. Aqui, foi utilizado para armazenar e manipular dados dos metais de forma estruturada em formato tabular.
- **Matplotlib**: é uma biblioteca de plotagem 2D que permite criar gráficos e visualizações de dados.
- **Math**: A biblioteca math oferece funções matemáticas básicas.

Código em Python

```python
import tkinter as tk
from tkinter import ttk, messagebox
import pandas as pd
import matplotlib.pyplot as plt
import math

class CorrosaoMonitoramento:
    metais = {
        'aço-carbono': {'densidade': 7.85, 'area_padrao': 20},
        'alumínio': {'densidade': 2.70, 'area_padrao': 20},
        'cobre': {'densidade': 8.96, 'area_padrao': 20},
        'zinco': {'densidade': 7.14, 'area_padrao': 20},
        'ferro': {'densidade': 7.87, 'area_padrao': 20},
        'latão': {'densidade': 8.44, 'area_padrao': 20},
        'níquel': {'densidade': 8.90, 'area_padrao': 20},
        'titânio': {'densidade': 4.51, 'area_padrao': 20}
    }
```

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

```
def __init__(self, metal):
    if metal in self.metais:
        self.densidade = self.metais[metal]['densidade']
        self.area_padrao = self.metais[metal]['area_padrao']
        self.metal = metal
    else:
        raise ValueError("Metal não encontrado na biblioteca.")
    self.dados = []

def calcular_taxa_corrosao(self, massa_inicial, massa_final, area, tempo, salinidade=0):
    K = 8.76e4
    perda_massa = massa_inicial - massa_final
    taxa_corrosao = (K * perda_massa) / (self.densidade * area * tempo) * (1 + salinidade / 100)
    return taxa_corrosao
```

```
def classificar_taxa_corrosao(self, taxa_corrosao):
    if taxa_corrosao < 0.1:
        return "Baixa"
    elif 0.1 <= taxa_corrosao < 0.5:
        return "Moderada"
    else:
        return "Alta"

def sugerir_solucao(self, classificacao):
    if classificacao == "Baixa":
        return "Solução sugerida para Baixa taxa de corrosão: Mor
    elif classificacao == "Moderada":
        return "Solução sugerida para Moderada taxa de corrosão:
    elif classificacao == "Alta":
        return "Solução sugerida para Alta taxa de corrosão: Apli
```

```
def calcular_pH_saturacao(self, STD, temp_C, Ca_CaCO3, alcalinidade_CaCO3):
    A = (math.log10(STD) - 1) / 10
    B = -13.12 * math.log10(temp_C + 273) + 34.55
    C = math.log10(Ca_CaCO3) - 0.4
    D = math.log10(alcalinidade_CaCO3)
    pHS = (9.3 + A + B) - (C + D)
    return pHS

def calcular_indice_langelier(self, pH, pHS):
    ISL = pH - pHS
    return ISL

def calcular_indice_ryznar(self, pH, pHS):
    IER = 2 * pHS - pH
    return IER
```

```
def classificar_ier(self, IER):
    if IER < 5.5:
        return "Intensa formação de incrustações"
    elif 5.5 <= IER < 6.2:
        return "Formação de incrustação"
    elif 6.2 <= IER < 6.8:
        return "Sem dificuldades"
    elif 6.8 <= IER < 8.5:
        return "Água agressiva"
    else:
        return "Água muito agressiva"
```

```
def adicionar_dados(self, data, massa_inicial, massa_final, area, tempo, salinidade, meio_aquoso, STD
    if meio_aquoso == 'sim':
        pHS = self.calcular_pH_saturacao(STD, temp_C, Ca_CaCO3, alcalinidade_CaCO3)
        ISL = self.calcular_indice_langelier(pH, pHS)
        IER = self.calcular_indice_ryznar(pH, pHS)
        classificacao_ier = self.classificar_ier(IER)
    else:
        pHS = ISL = IER = classificacao_ier = None

    taxa_corrosao = self.calcular_taxa_corrosao(massa_inicial, massa_final, area, tempo, salinidade)
    classificacao = self.classificar_taxa_corrosao(taxa_corrosao)
    solucao = self.sugerir_solucao(classificacao)
```

```
self.dados.append({
    'data': data,
    'massa_inicial': massa_inicial,
    'massa_final': massa_final,
    'area': area,
    'tempo': tempo,
    'salinidade': salinidade,
    'meio_aquoso': meio_aquoso,
    'taxa_corrosao': taxa_corrosao,
    'classificacao': classificacao,
    'solucao': solucao,
    'pH': pH if meio_aquoso == 'sim' else None,
    'pHS': pHS,
    'ISL': ISL,
    'IER': IER,
    'classificacao_ier': classificacao_ier
})
```

```python
def gerar_grafico(self):
    if not self.dados:
        messagebox.showwarning("Atenção", "Nenhum dado adicionado. Adicione dados primeiro.")
        return

    df = pd.DataFrame(self.dados)
    df['data'] = pd.to_datetime(df['data'])
    df.set_index('data', inplace=True)
    df['taxa_corrosao'].plot(marker='o', linestyle='-')
    plt.title(f'Taxa de Corrosão ao Longo do Tempo ({self.metal.capitalize()})')
    plt.xlabel('Data')
    plt.ylabel('Taxa de Corrosão (mm/ano)')
    plt.grid(True)
    plt.show()
```

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

```
    def imprimir_dados(self):
        if not self.dados:
            messagebox.showwarning("Atenção", "Nenhum dado adicionado. Adicione dados primeiro.")
            return

        df = pd.DataFrame(self.dados)
        print(df)

        for index, row in df.iterrows():
            print(f"\nSolução sugerida para a data {index}:")
            print(row['solucao'])
            print("=" * 50)

class AplicacaoCorrosao:
    def __init__(self, root):
        self.root = root
        self.root.title("Monitoramento de Corrosão")
        self.root.geometry('800x600')  # Defina o tamanho da janela
```

```
self.monitoramento = None  # A instância do monitoramento será armazenada aqui

# Frame para os campos de entrada
self.frame_entrada = ttk.Frame(root)
self.frame_entrada.pack(padx=10, pady=10)

self.metal_label = ttk.Label(self.frame_entrada, text="Escolha um metal:")
self.metal_label.grid(row=0, column=0, padx=5, pady=5)
self.metal_var = tk.StringVar()
self.metal_combobox = ttk.Combobox(self.frame_entrada, textvariable=self.metal_var)
self.metal_combobox['values'] = ('aço-carbono', 'alumínio', 'cobre', 'zinco', 'ferro'
self.metal_combobox.grid(row=0, column=1, padx=5, pady=5)
```

```
self.data_label = ttk.Label(self.frame_entrada, text="Data (YYYY-MM-DD):")
self.data_label.grid(row=1, column=0, padx=5, pady=5)
self.data_entry = ttk.Entry(self.frame_entrada)
self.data_entry.grid(row=1, column=1, padx=5, pady=5)

self.massa_inicial_label = ttk.Label(self.frame_entrada, text="Massa Inicial (g):")
self.massa_inicial_label.grid(row=2, column=0, padx=5, pady=5)
self.massa_inicial_entry = ttk.Entry(self.frame_entrada)
self.massa_inicial_entry.grid(row=2, column=1, padx=5, pady=5)

self.massa_final_label = ttk.Label(self.frame_entrada, text="Massa Final (g):")
self.massa_final_label.grid(row=3, column=0, padx=5, pady=5)
self.massa_final_entry = ttk.Entry(self.frame_entrada)
self.massa_final_entry.grid(row=3, column=1, padx=5, pady=5)
```

```
self.area_label = ttk.Label(self.frame_entrada, text="Área (cm²):")
self.area_label.grid(row=4, column=0, padx=5, pady=5)
self.area_entry = ttk.Entry(self.frame_entrada)
self.area_entry.grid(row=4, column=1, padx=5, pady=5)

self.tempo_label = ttk.Label(self.frame_entrada, text="Tempo (horas):")
self.tempo_label.grid(row=5, column=0, padx=5, pady=5)
self.tempo_entry = ttk.Entry(self.frame_entrada)
self.tempo_entry.grid(row=5, column=1, padx=5, pady=5)

self.salinidade_label = ttk.Label(self.frame_entrada, text="Salinidade (%):")
self.salinidade_label.grid(row=6, column=0, padx=5, pady=5)
self.salinidade_entry = ttk.Entry(self.frame_entrada)
self.salinidade_entry.grid(row=6, column=1, padx=5, pady=5)
```

```
self.meio_aquoso_label = ttk.Label(self.frame_entrada, text="Meio Aquoso (sim/não):")
self.meio_aquoso_label.grid(row=7, column=0, padx=5, pady=5)
self.meio_aquoso_var = tk.StringVar()
self.meio_aquoso_combobox = ttk.Combobox(self.frame_entrada, textvariable=self.meio_aquoso_v
self.meio_aquoso_combobox['values'] = ('sim', 'não')
self.meio_aquoso_combobox.grid(row=7, column=1, padx=5, pady=5)

self.STD_label = ttk.Label(self.frame_entrada, text="STD:")
self.STD_label.grid(row=8, column=0, padx=5, pady=5)
self.STD_entry = ttk.Entry(self.frame_entrada)
self.STD_entry.grid(row=8, column=1, padx=5, pady=5)

self.temp_C_label = ttk.Label(self.frame_entrada, text="Temperatura (°C):")
self.temp_C_label.grid(row=9, column=0, padx=5, pady=5)
self.temp_C_entry = ttk.Entry(self.frame_entrada)
self.temp_C_entry.grid(row=9, column=1, padx=5, pady=5)
```

```
self.Ca_CaCO3_label = ttk.Label(self.frame_entrada, text="Cálcio (CaCO3) (mg/L):")
self.Ca_CaCO3_label.grid(row=10, column=0, padx=5, pady=5)
self.Ca_CaCO3_entry = ttk.Entry(self.frame_entrada)
self.Ca_CaCO3_entry.grid(row=10, column=1, padx=5, pady=5)

self.alcalinidade_CaCO3_label = ttk.Label(self.frame_entrada, text="Alcalinidade (CaCO3) (mg/L):
self.alcalinidade_CaCO3_label.grid(row=11, column=0, padx=5, pady=5)
self.alcalinidade_CaCO3_entry = ttk.Entry(self.frame_entrada)
self.alcalinidade_CaCO3_entry.grid(row=11, column=1, padx=5, pady=5)

self.pH_label = ttk.Label(self.frame_entrada, text="pH:")
self.pH_label.grid(row=12, column=0, padx=5, pady=5)
self.pH_entry = ttk.Entry(self.frame_entrada)
self.pH_entry.grid(row=12, column=1, padx=5, pady=5)
```

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

```
    self.adicionar_button = ttk.Button(self.frame_entrada, text="Adicionar Dados", command=self.adicionar_dados)
    self.adicionar_button.grid(row=13, column=0, columnspan=2, pady=10)

    self.gerar_grafico_button = ttk.Button(self.frame_entrada, text="Gerar Gráfico", command=self.gerar_grafico)
    self.gerar_grafico_button.grid(row=14, column=0, columnspan=2, pady=10)

    self.imprimir_dados_button = ttk.Button(self.frame_entrada, text="Imprimir Dados", command=self.imprimir_dados)
    self.imprimir_dados_button.grid(row=15, column=0, columnspan=2, pady=10)

def adicionar_dados(self):
    metal = self.metal_var.get()
    if not self.monitoramento:
        self.monitoramento = CorrosaoMonitoramento(metal)
    data = self.data_entry.get()
    massa_inicial = float(self.massa_inicial_entry.get())
    massa_final = float(self.massa_final_entry.get())
```

```
area = float(self.area_entry.get())
tempo = float(self.tempo_entry.get())
salinidade = float(self.salinidade_entry.get())
meio_aquoso = self.meio_aquoso_var.get()
STD = float(self.STD_entry.get()) if meio_aquoso == 'sim' else None
temp_C = float(self.temp_C_entry.get()) if meio_aquoso == 'sim' else None
Ca_CaCO3 = float(self.Ca_CaCO3_entry.get()) if meio_aquoso == 'sim' else None
alcalinidade_CaCO3 = float(self.alcalinidade_CaCO3_entry.get()) if meio_aquoso == 'sim' else None
pH = float(self.pH_entry.get()) if meio_aquoso == 'sim' else None

self.monitoramento.adicionar_dados(data, massa_inicial, massa_final, area, tempo, salinidade, meio_a
messagebox.showinfo("Sucesso", "Dados adicionados com sucesso!")
```

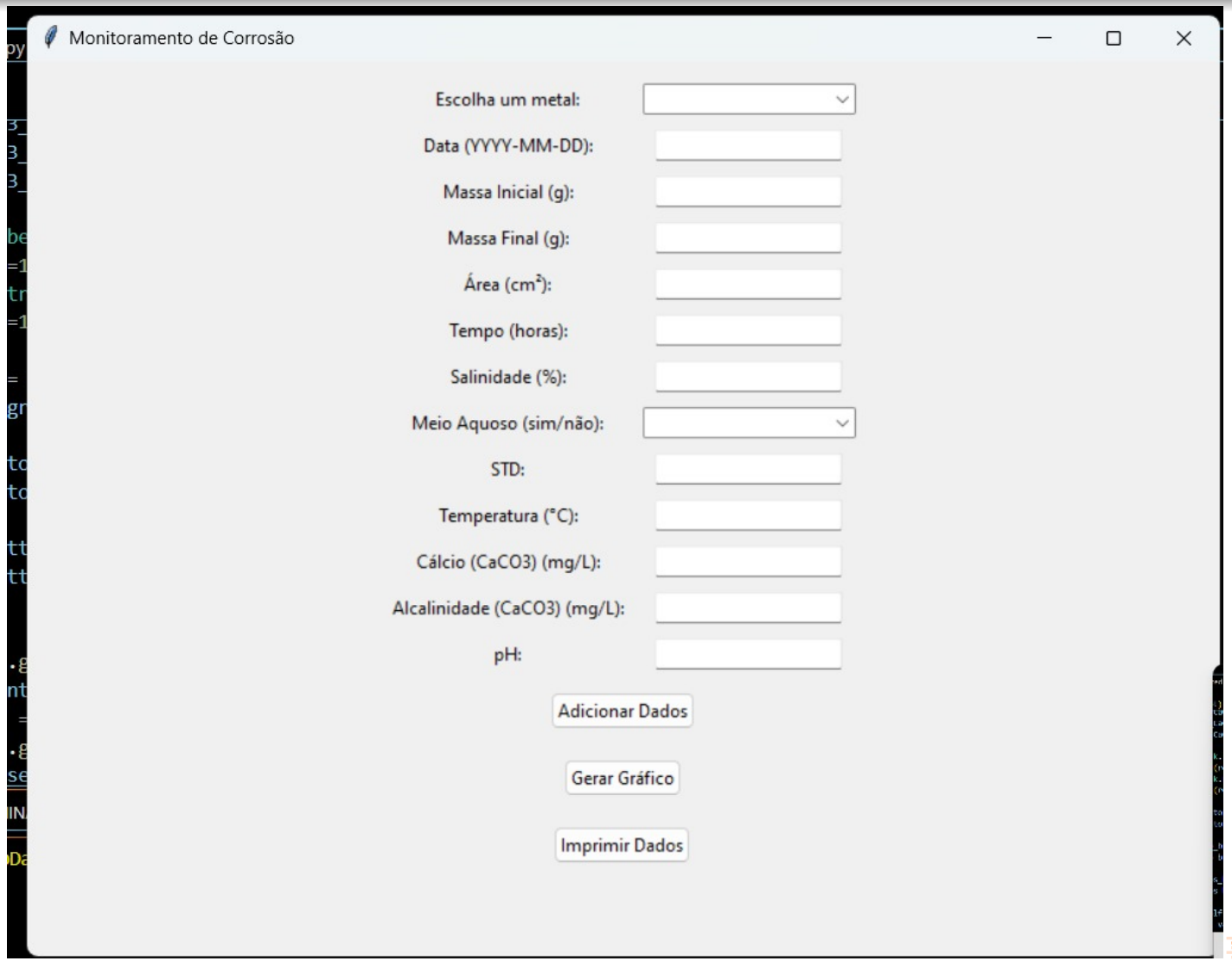
```
    def gerar_grafico(self):
        if self.monitoramento:
            self.monitoramento.gerar_grafico()
        else:
            messagebox.showwarning("Atenção", "Nenhum dado adicionado. Adicione dados primeiro.")

    def imprimir_dados(self):
        if self.monitoramento:
            self.monitoramento.imprimir_dados()
        else:
            messagebox.showwarning("Atenção", "Nenhum dado adicionado. Adicione dados primeiro.")

if __name__ == "__main__":
    root = tk.Tk()
    app = AplicacaoCorrosao(root)
    root.mainloop()
```

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Sumário

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Amostras	0	1	2	3	4
Data	15/06/19	15/06/20	15/06/21	15/06/22	15/06/23
Massa inicial	100	110	95	105	115
Massa final	98	105	92	100	110
Área (cm^2)	25	30	22	28	32
Tempo (horas)	24	24	24	24	24
Salinidade (%)	1,5	0,8	1,0	0,5	1,2
Meio Aquoso	sim	sim	sim	sim	sim
STD	200	180	190	170	210
Temperatura (°C)	25	22	23	20	26
Cálcio (CaCO3) (mg/L)	50	40	45	35	55
Alcalinidade (CaCO3) (mg/L)	80	70	75	65	85
PH	7,5	7,2	7,0	6,8	7,3



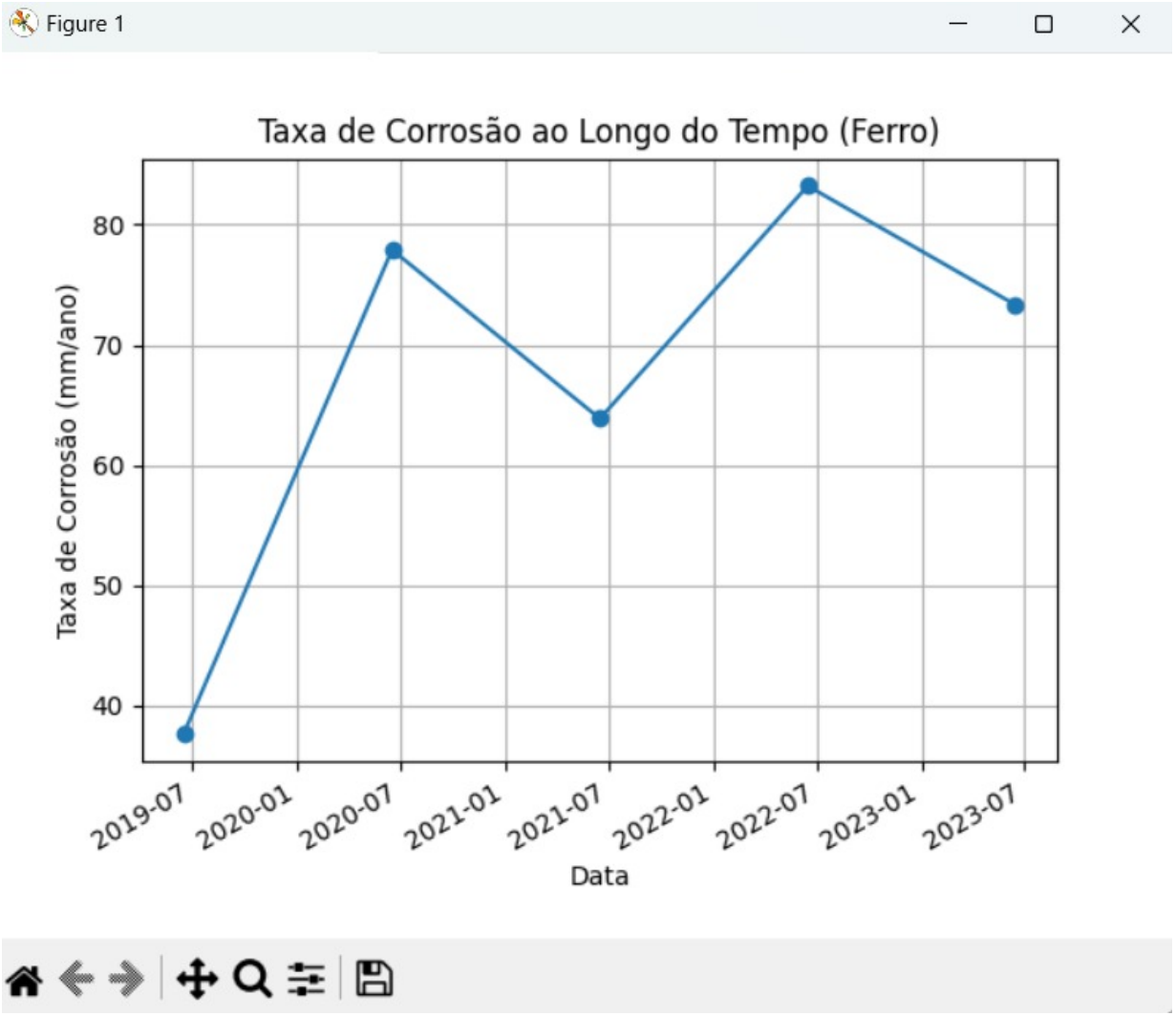
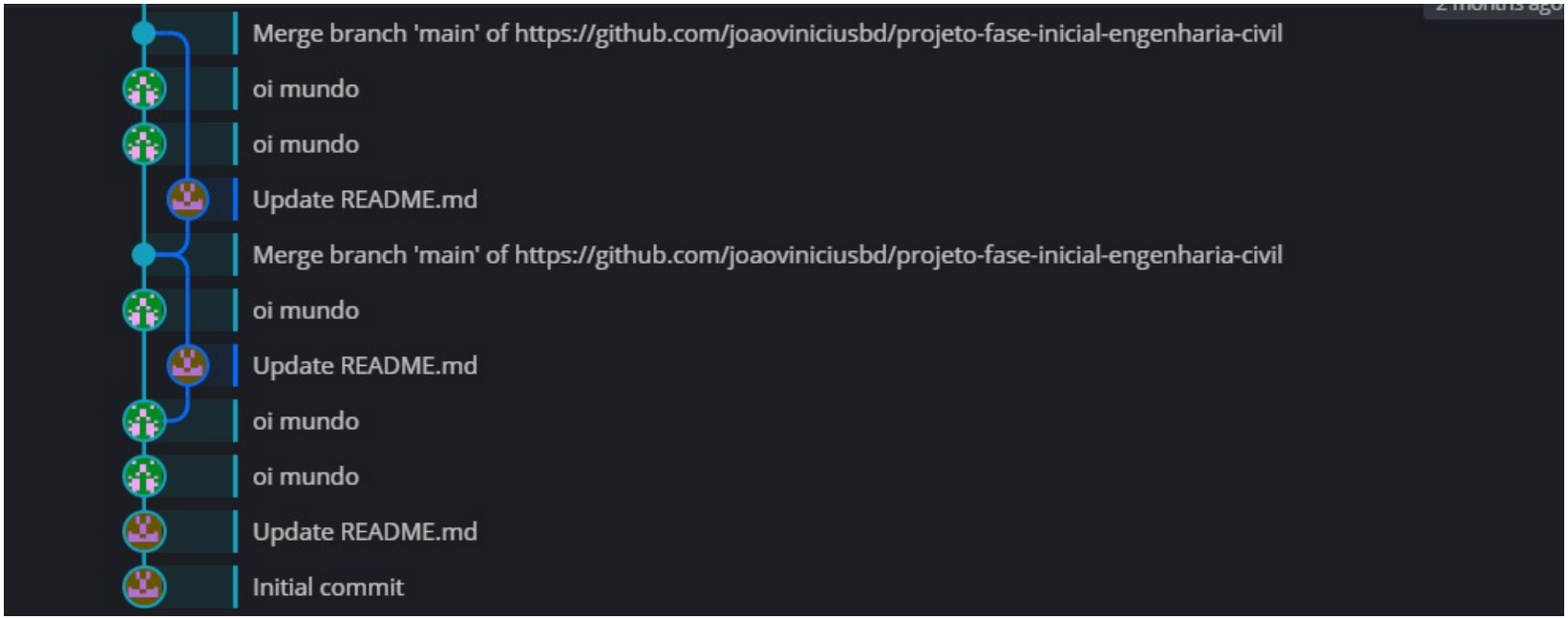


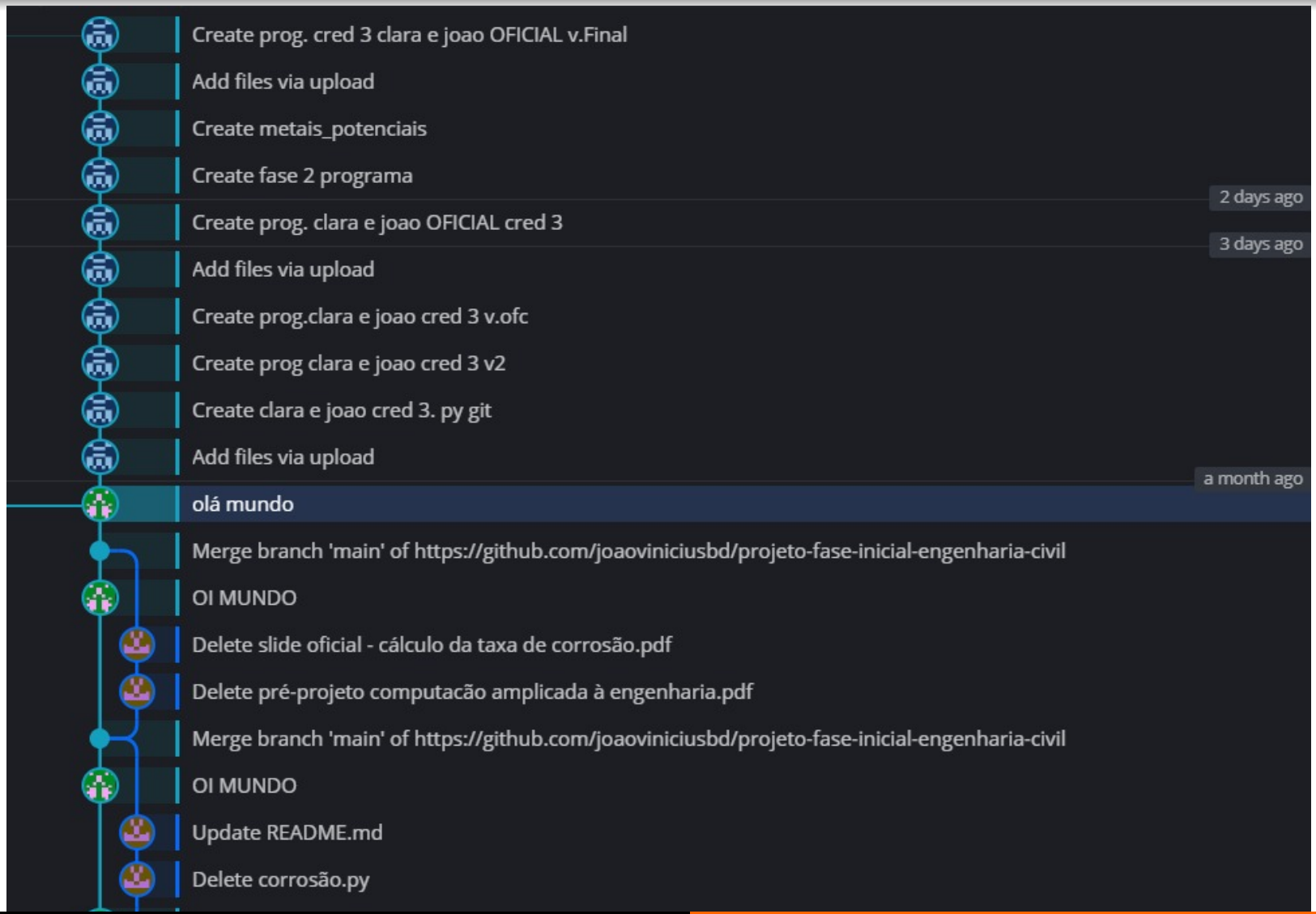
Figure 1
Taxa de Corrosão ao Longo do Tempo (Ferro)
Taxa de Corrosão (mm/ano)
80
70
60
50
40
2019-07
2020-01
2020-07
2021-01
2021-07
2022-01
2022-07
2023-01
2023-07
Data

```
PS C:\Users\clara> & C:/Users/clara/AppData/Local/Microsoft/WindowsApps/python3.12.exe "c:/python/prog clara e joao oficial c
         data  massa_inicial  massa_final  area  tempo  ...   pH       pHS       ISL        IER     classificacao_ier
0  2019-06-15          100.0         98.0  25.0   24.0  ...  7.5  8.316326 -0.816326   9.132651  Água muito agressiva
1  2020-06-15          110.0        115.0  30.0   24.0  ...  7.2  8.524304 -1.324304   9.848609  Água muito agressiva
2  2021-06-15           95.0         92.0  22.0   24.0  ...  7.0  8.426254 -1.426254   9.852509  Água muito agressiva
3  2022-06-15          105.0        100.0  28.0   24.0  ...  6.8  8.650760 -1.850760  10.501521  Água muito agressiva
4  2023-06-15          115.0        110.0  32.0   24.0  ...  7.3  8.231634 -0.931634   9.163269  Água muito agressiva

[5 rows x 15 columns]

Solução sugerida para a data 0:
Solução sugerida para Alta taxa de corrosão: Aplicação de revestimentos mais resistentes e revisão do ambiente de exposição.
================================================

Solução sugerida para a data 1:
Solução sugerida para Baixa taxa de corrosão: Monitoramento regular e manutenção preventiva.
================================================

Solução sugerida para a data 2:
Solução sugerida para Alta taxa de corrosão: Aplicação de revestimentos mais resistentes e revisão do ambiente de exposição.
Solução sugerida para a data 3:
Solução sugerida para Alta taxa de corrosão: Aplicação de revestimentos mais resistentes e revisão do ambiente de exposição.
================================================

Solução sugerida para a data 4:
Solução sugerida para Alta taxa de corrosão: Aplicação de revestimentos mais resistentes e revisão do ambiente de exposição.
================================================
```

Sumário

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA


Merge branch 'main' of https://github.com/joaoviniciusbd/projeto-fase-inicial-engenharia-civil
oi mundo
oi mundo
Update README.md
Merge branch 'main' of https://github.com/joaoviniciusbd/projeto-fase-inicial-engenharia-civil
oi mundo
Update README.md
oi mundo
oi mundo
Update README.md
Initial commit


Create prog. cred 3 clara e joao OFICIAL v.Final
Add files via upload
Create metais_potenciais
Create fase 2 programa
2 days ago
Create prog. clara e joao OFICIAL cred 3
3 days ago
Add files via upload
Create prog.clara e joao cred 3 v.ofc
Create prog clara e joao cred 3 v2
Create clara e joao cred 3. py git
Add files via upload
a month ago
olá mundo
Merge branch 'main' of https://github.com/joaoviniciusbd/projeto-fase-inicial-engenharia-civil
OI MUNDO
Delete slide oficial - cálculo da taxa de corrosão.pdf
Delete pré-projeto computacão amplicada à engenharia.pdf
Merge branch 'main' of https://github.com/joaoviniciusbd/projeto-fase-inicial-engenharia-civil
OI MUNDO
Update README.md
Delete corrosão.py

Sumário

O programa desenvolvido oferece uma solução robusta e prática para enfrentar um problema comum e oneroso em diversas indústrias, a corrosão. Este sistema proporciona uma abordagem abrangente para o monitoramento da corrosão em diferentes metais e a análise da água, integrando um conjunto de funcionalidades. Além de calcular a taxa de corrosão, o programa classifica a sua intensidade em categorias claramente definidas e compreensíveis. Com base nessa classificação, são sugeridas soluções práticas, adaptadas ao nível de severidade da corrosão, tornando-se, assim, uma ferramenta indispensável na manutenção e preservação de materiais metálicos em diversos ambientes. Engenheiros civis podem utilizar esses dados para selecionar metais que ofereçam maior resistência à corrosão, otimizando os recursos disponíveis e reduzindo os custos associados à manutenção corretiva.

Sumário

UNEB
UNIVERSIDADE DO ESTADO DA BAHIA

Referências

- LTC Gentil, V. editora, 5ª edição. Rio de Janeiro, 2007.
- William Callister. Ciência E Engenharia de Materiais: Uma Introdução. Grupo Gen-LTC, 2000.
- Stephan Wolynec. Técnicas eletroquímicas em corrosão Vol. 49. Edusp, 2003.
- Nilo Ney Coutinho Menezes. Introdução à programação com Python. São Paulo: Novatec, página 34, 2010.
- Felipe Cruz. Python: Escreva seus primeiros programas. Editora Casa do Código, 2015.