APLICAÇÃO DE DESIGN DE SOFTWARE + DIAGRAMA UML DO PROJETO

ENTREGA 2.0

Introdução

Este documento apresenta a aplicação dos princípios de design de software e o diagrama UML utilizado no Projeto Interdisciplinar. Analisamos o arquivo app.py, que monta um Painel de Indicadores Econômicos, desenvolvido com Streamlit e integração com a API do Banco Central (BCB).

Design de Software Aplicado

O projeto adota princípios de orientação a objetos e separação de responsabilidades, organizando as funcionalidades em classes específicas:

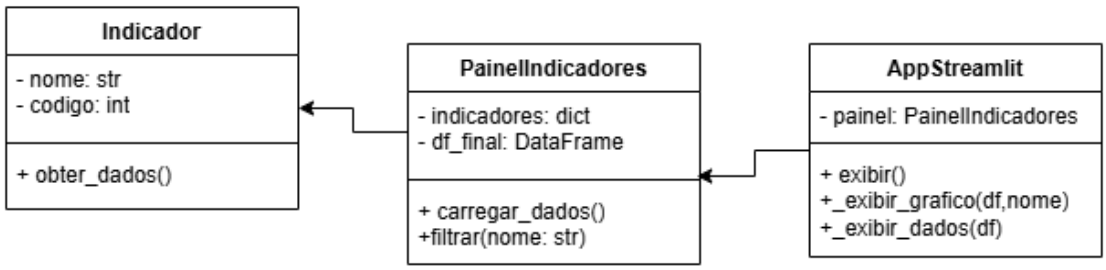
- **Indicador**: Responsável por representar um indicador econômico (ex: Selic, IPCA) e obter seus dados via API.
- **PainelIndicadores**: Gerencia o conjunto de indicadores, realizando a carga e filtragem dos dados.
- **AppStreamlit**: Responsável pela interface gráfica usando Streamlit. Apresenta os gráficos e dados ao usuário.

Princípios de design aplicados:

Princípio	Aplicação
Encapsulamento	As classes encapsulam comportamentos e dados relacionados a suas responsabilidades.
Responsabilidade Única (SRP)	Cada classe tem uma única responsabilidade.
Abstração	Os detalhes de obtenção de dados estão abstraídos dentro da classe Indicador.
Separação de camadas	Lógica de dados e lógica de apresentação estão separadas.

Diagrama de Classes UML

A seguir, apresentamos o **diagrama de classes UML** representando a estrutura principal do projeto:



Código respeitando os princípios do design de software:

Abaixo, apresentamos o **código do painel de indicadores econômicos**:

```
import streamlit as st
import pandas as pd
import plotly.express as px
from bcb import sgs

# --- CLASSES DO PROJETO ---

class Indicador:
    def __init__(self, nome, codigo):
        self.nome = nome
        self.codigo = codigo

    def obter_dados(self, inicio='2020-01-01'):
        df = sgs.get(self.codigo, start=inicio)
        df.reset_index(inplace=True)
        df.rename(columns={'index': 'Data', self.nome: 'Valor'},
inplace=True)
        df['Indicador'] = self.nome
        return df


class PainelIndicadores:
    def __init__(self):
        self.indicadores = {
            'Selic': 11,
            'IPCA': 4449,
            'IGP-M': 189,
            'Dólar': 1
        }
        self.df_final = None

    def carregar_dados(self):
        dados = []
        for nome, codigo in self.indicadores.items():
```

```
            indicador = Indicador(nome, codigo)
            dados.append(indicador.obter_dados())
        self.df_final = pd.concat(dados)

    def filtrar(self, nome_indicador):
        return self.df_final[self.df_final['Indicador'] ==
nome_indicador]


class AppStreamlit:
    def __init__(self, painel):
        self.painel = painel

    def exibir(self):
        st.set_page_config(layout='wide')
        st.title(' Painel de Indicadores Econômicos')
        st.write('Fonte: Banco Central do Brasil (BCB)')

        indicadores = self.painel.df_final['Indicador'].unique()
        indicador_escolhido = st.selectbox('Escolha um indicador:',
indicadores)

        df_filtrado = self.painel.filtrar(indicador_escolhido)

        self._exibir_grafico(df_filtrado, indicador_escolhido)
        self._exibir_dados(df_filtrado)

    def _exibir_grafico(self, df, nome):
        fig = px.line(
            df, x='Data', y='Valor',
            title=f'Evolução do {nome}',
            labels={'Valor': 'Valor', 'Data': 'Data'},
            template='plotly_dark'
        )
        st.plotly_chart(fig, use_container_width=True)

    def _exibir_dados(self, df):
        st.write(' **Dados Brutos:**')
        st.dataframe(df)


# --- EXECUÇÃO DO APP ---

if __name__ == '__main__':
    painel = PainelIndicadores()
    painel.carregar_dados()

    app_ui = AppStreamlit(painel)
    app_ui.exibir()
```

Considerações Finais

O uso da orientação a objetos e princípios de design de software contribuiu para a organização, reutilização e manutenibilidade do código. O diagrama UML facilita o entendimento da estrutura do sistema, servindo como uma base para documentação e futuras expansões.