Design de Software e Modelagem UML

Design de Software

1.1. Visão Geral do Sistema

O sistema a ser desenvolvido é um **demonstrativo preditivo interativo** que utiliza dados macroeconômicos do **Banco Central do Brasil** para realizar predições sobre indicadores econômicos como a **taxa Selic**, **IPCA** e **câmbio**, entre outros. O objetivo é fornecer uma plataforma interativa que permita aos usuários visualizar e analisar esses dados, identificando padrões e realizando predições baseadas em algoritmos de **Machine Learning**.

O sistema será composto por dois principais componentes:

- **Backend**: Responsável pela coleta de dados, processamento e execução das predições.
- **Frontend**: Interface interativa onde os usuários poderão visualizar os dados e interagir com as predições.

1.2. Arquitetura do Sistema

A arquitetura do sistema será baseada em uma **arquitetura de três camadas** (ou seja, **separação de responsabilidades**):

1. **Camada de Apresentação (Frontend)**:
    a. O frontend será desenvolvido utilizando **React** com **Next.js**, que permite a criação de interfaces ricas e rápidas. O **Next.js** facilitará o desenvolvimento de uma aplicação escalável e otimizada, com suporte a SSR (Server-Side Rendering) para melhor performance.
    b. O objetivo é criar um **dashboard interativo** que permitirá aos usuários visualizar dados históricos e predições, além de realizar interações como filtros e comparações de dados.
2. **Camada de Lógica de Negócio (Backend)**:
    a. O backend será desenvolvido em **Python**, utilizando as bibliotecas **Pandas** e **Numpy** para o processamento de dados. **Scikit-Learn** será utilizado para a parte de **Machine Learning**, permitindo realizar predições com base em dados históricos.
    b. A comunicação entre o backend e o frontend será realizada por meio de uma **API RESTful**, desenvolvida com **FastAPI** ou **Flask**. Essa API fornecerá endpoints para a obtenção dos dados do Banco Central e para realizar as predições de acordo com os dados fornecidos.
3. **Camada de Persistência (Banco de Dados)**:

a. O banco de dados escolhido é o **MongoDB**, devido à sua flexibilidade em lidar com grandes volumes de dados e sua facilidade de escalabilidade. O MongoDB será utilizado para armazenar dados históricos obtidos da API do Banco Central e os resultados das predições.

1.3. Tecnologias e Ferramentas Utilizadas

- **Frontend**:
  - **React**: Biblioteca JavaScript para construção de interfaces de usuário.
  - **Next.js**: Framework React para renderização do lado servidor e otimização de performance.
  - **Chart.js ou D3.js** (opcional): Bibliotecas para visualização gráfica dos dados no frontend.
- **Backend**:
  - **Python**: Linguagem de programação principal para manipulação de dados e criação de modelos de Machine Learning.
  - **Pandas** e **Numpy**: Bibliotecas para manipulação e análise de dados.
  - **Scikit-Learn**: Biblioteca de Machine Learning para construir e treinar modelos preditivos.
  - **FastAPI** ou **Flask**: Frameworks para criação da API RESTful que comunica o backend com o frontend.
- **Banco de Dados**:
  - **MongoDB**: Banco de dados NoSQL, ideal para armazenar dados não estruturados e grandes volumes de dados.

1.4. Fluxo de Dados no Sistema

1. **Coleta de Dados**:
   a. O sistema irá coletar os dados do **Banco Central do Brasil** por meio de sua API pública. Os dados coletados incluem indicadores como a **taxa Selic**, **IPCA** e outros índices econômicos importantes.
   b. Esses dados serão armazenados no banco de dados MongoDB para posterior análise e predição.
2. **Processamento e Análise**:
   a. Os dados coletados serão processados no backend utilizando **Pandas** e **Numpy** para limpeza e organização.
   b. O **Scikit-Learn** será utilizado para construir e treinar modelos preditivos com base nos dados históricos coletados. Esses modelos poderão prever tendências futuras para os indicadores econômicos.
3. **Exibição e Interação**:
   a. No frontend, o usuário poderá visualizar os dados históricos e as predições geradas pelo backend.

b. O **dashboard** interativo permitirá ao usuário filtrar os dados, ajustar parâmetros e visualizar gráficos interativos dos resultados.

1.5. Modelo de Segurança e Escalabilidade

Para garantir a **segurança** e **escalabilidade** do sistema:

- O **FastAPI** ou **Flask** será configurado com autenticação básica (caso seja necessário) para controlar o acesso aos dados sensíveis.
- O **MongoDB** será configurado para suportar grandes volumes de dados e permitir uma escalabilidade horizontal conforme a necessidade de crescimento do sistema.

1.6. Justificativa das Escolhas Tecnológicas

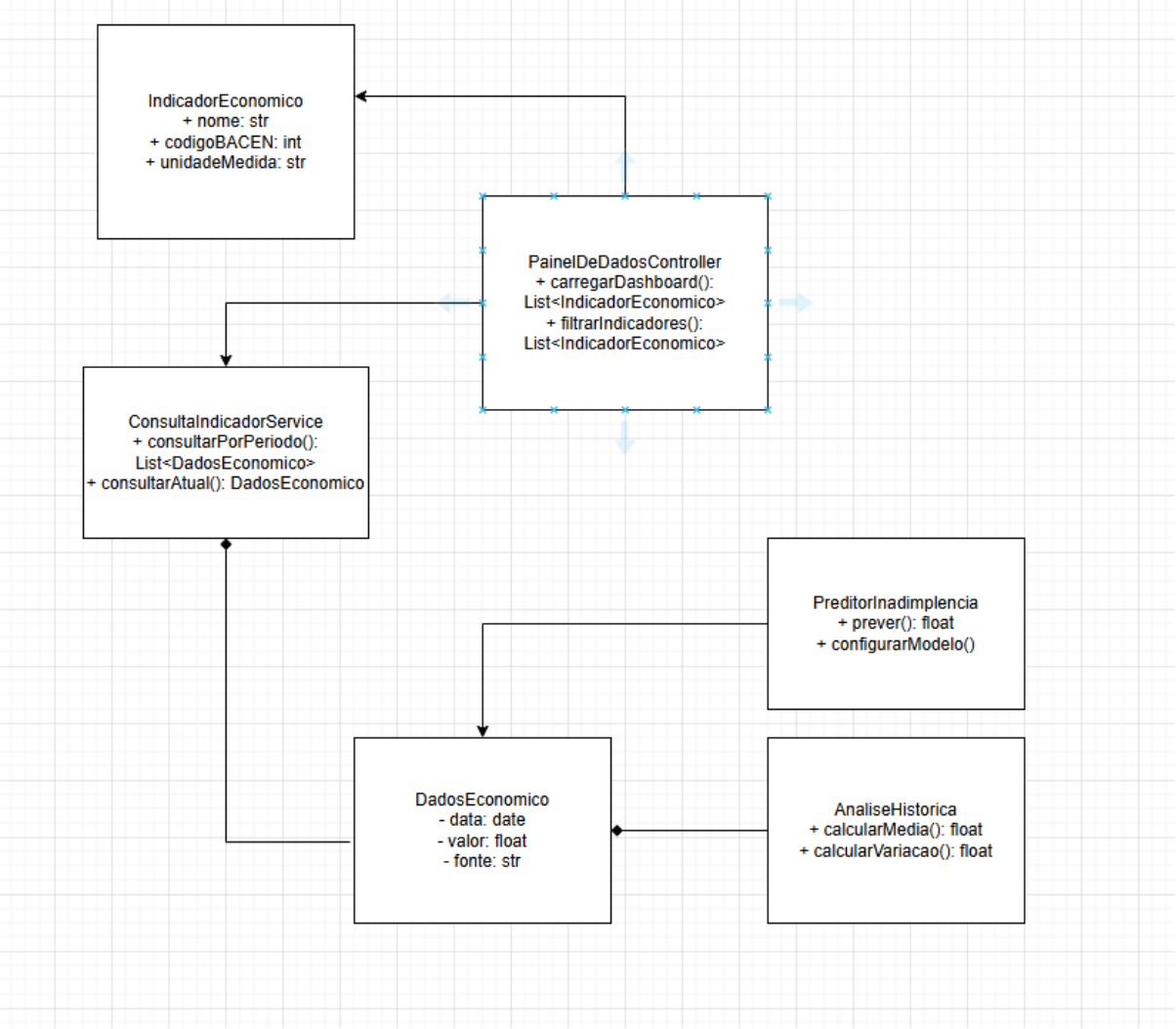
A escolha das tecnologias foi baseada nos seguintes critérios:

- **React + Next.js** foi escolhido pelo seu ecossistema robusto e flexibilidade no desenvolvimento de interfaces de usuário dinâmicas e rápidas.
- **Python** é a melhor linguagem para análise de dados e Machine Learning devido à sua ampla gama de bibliotecas e facilidade de integração com outras tecnologias.
- **MongoDB** foi escolhido pela sua flexibilidade em lidar com dados não estruturados e pela facilidade de escalabilidade.

Diagrama de Classe - UML

A seguir, apresentamos o **diagrama de classes UML** do sistema, representando a estrutura e as responsabilidades principais de cada classe envolvida no projeto.

Diagrama de Classes



Descrição Geral das Classes

- **IndicadorEconomico**: Representa um indicador, como IPCA ou Selic, contendo nome, código do BACEN e unidade de medida.
- **DadosEconomico**: Armazena os dados históricos dos indicadores, com data, valor e fonte.
- **ConsultaIndicadorService**: Responsável por consultar os dados dos indicadores, seja atual ou por período.
- **PainelDeDadosController**: Controlador principal que carrega os dados no dashboard e permite filtrar os indicadores.
- **PreditorInadimplencia**: Classe dedicada ao modelo preditivo, responsável por prever a inadimplência com base nos dados.
- **AnaliseHistorica**: Realiza cálculos estatísticos como média e variação dos dados históricos.