Desafio de Backend - Engenheiro(a) de Software

Introdução

Olá! Seja bem-vindo(a) ao nosso desafio para Engenheiro(a) de Software backend!

Em nosso dia a dia trabalhamos com um volume massivo de dados transacionais, onde a performance e a escalabilidade não são apenas diferenciais, mas requisitos fundamentais. Este desafio foi projetado para simular um desses cenários, avaliando sua capacidade de projetar e construir uma solução de processamento de dados que seja, acima de tudo, eficiente e robusta.

Queremos observar como você estrutura seu código, modela os dados e, principalmente, como otimiza o processamento para lidar com uma carga de trabalho intensa, um problema comum no mercado financeiro.

Desafio

Seu objetivo é desenvolver uma aplicação que realize a ingestão, o processamento e a exposição de dados agregados de negociações da B3.

Fonte dos Dados e Formato

Os dados brutos podem ser obtidos no site da B3. Para este desafio, focaremos nos arquivos de negociações de ações (Stocks).

- **Link:** Cotações [Cotações | B3]
- **Estrutura do Arquivo:** Você pode assumir que a estrutura e a ordem das colunas do CSV serão fixas.
- **Campos Relevantes:** Para este desafio, você precisará dos seguintes atributos
    - **`DataNegocio:`** Data que o negócio ocorreu.
    - **`CodigoInstrumento:`** "ticker" do ativo, ex: "PETR4".
    - **`PrecoNegocio:`** 10,000 valor unitário do ativo.
    - **`QuantidadeNegociada:`** quantidade de ativos da negociação.
    - **`HoraFechamento:`** (formato HHMMSSmmm). Horário da negociação.

Requisitos Funcionais

1. **Ingestão de dados:**
    - **Escopo:** A aplicação deve processar os dados históricos de negociações da B3 referentes aos últimos **7 dias úteis.**
    - **Mecanismo de Carga:**
        - Você pode baixar os arquivos CSV manualmente e disponibilizá-los em um diretório local para serem processados pela sua aplicação.
        - O foco da avaliação **não está** na automação do download, e sim **na eficiência do pipeline de ingestão e processamento** que você irá construir através de uma aplicação (CLI ou similar). Durante a avaliação nós iremos seguir suas instruções no arquivo `README.md` a risca.
    - **Requisito de Performance de Ingestão:** Dado o grande volume de dados, a ingestão completa dos 7 dias de arquivos deve ser concluída em um tempo razoável. O seu processo de carga deve ser executado em menos de 15 minutos em uma máquina de desenvolvimento padrão (ex: Docker em um notebook com 16GB de RAM, 6 cores). Estratégias de concorrência, batching, gerenciamento de memória e modelagem dos dados serão pontos-chave da para o sucesso desta etapa.

2. **Persistência de Dados:**
    - Os campos relevantes de cada negociação devem ser persistidos em um banco de dados de sua escolha (ex: PostgreSQL, MySQL, SQLite, MonogoDB, BoltDB, BadgerDB, etc). A modelagem do schema e a escolha dos tipos de dados são parte da avaliação.

3. **Interface de Consulta:**
    - Você deve expor uma interface para consulta dos dados agregados através de uma API REST em um único endpoint.
    - **A interface deve aceitar os seguintes filtros:**
        - `ticker` (obrigatório): Ativo a ser analisado.
        - `data_inicio` (opcional): Somente a data no formato ISO-8601.

- **Regra do Filtro de Data:**
    - Se `data_inicio` for omitido, a consulta deve abranger os dados ingeridos nos últimos 7 dias, tendo como último dia do período de análise a data anterior à atual.
    - Se fornecido, a consulta deve incluir dados onde a `DataNegocio` é maior ou igual (>=) à data informada, até o final do período disponível.

- **Definição da Saída (JSON)**
    - A consulta deve retornar um objeto JSON com a seguinte estrutura:

```JSON
{
      ticker: "PETR4",
      max_range_value: 20.50,
      max_daily_volume: 150000
}
```

        - **`ticker:`** O código do ativo que foi utilizado no filtro.
        - **`max_range_value`**: O maior preço unitário (`PrecoNegocio`) registrado para o ticker em todo o período filtrado.
        - **`max_daily_volume`**: O volume máximo de ativos negociados (`QuantidadeNegociada`) consolidado em um único dia para o ticker.
            - Exemplo: Se para o ticker "VALE3", no dia 1 o volume total negociado **no dia** foi de 100.000 ações e no dia 2 o volume total foi de 150.000, o valor retornado para `max_daily_volume` deve ser 150.000.

Critérios de Avaliação

1. **Engenharia de Software e Performance**
    - **Performance de Ingestão:** Conforme o requisito funcional, analisaremos a eficiência da sua solução para carregar, processar e armazenar os dados dos arquivos. O uso de concorrência/paralelismo, gerenciamento de memória e estratégias para otimizar o I/O de disco e a modelagem e técnicas avançadas de banco de dados são fundamentais.
    - **Performance de Leitura:** A eficiência das consultas será testada. Esperamos que as agregações sejam rápidas (idealmente, com tempo de resposta preferencialmente sub-segundo, porém serão considerados os desafios que não cumpram este requisito), mesmo ao consultar tickers com altíssimo volume de negociação no período completo de 7 dias.

2. **Arquitetura e Modelagem de Dados**
    - **Modelagem de Banco de Dados:** A estrutura das tabelas, a escolha dos tipos de dados, índices e outras otimizações, técnicas e heurísticas que visem o equilíbrio entre performance de escrita (ingestão) e de leitura (consultas).
    - **Robustez e Confiabilidade:** Como a aplicação lida com a integridade dos dados. A idempotência é um critério relevante a ser considerado.

3. **Qualidade de Código e Testes**
    - **Clareza e Manutenibilidade:** Código limpo, bem organizado, documentado e que siga as práticas idiomáticas da linguagem escolhida também será considerado na avaliação.
    - **Testes:** A aplicação não precisa ter alta cobertura de testes, mas os testes existentes devem ser concisos, significativos e cobrir os caminhos críticos e alternativos da lógica de negócio.

4. **Reprodutibilidade (Guia de Entrega)**
    - **Documentação:** Um `README.md` claro e objetivo é essencial pois iremos segui-lo à risca. Ele deve explicar as decisões de arquitetura e, principalmente, conter instruções precisas para:
        i. Configurar o ambiente.
        ii. Construir a aplicação.
        iii. Executar a ingestão dos dados.
        iv. Executar a aplicação e as consultas.

        * Nosso ambiente é composto por máquinas com processadores Intel, Sistema Operacional Linux Ubuntu 22.04 LTS + Docker com Plugin Composer.

    - **Conteinerização:** A solução deve ser entregue com um `docker-compose.yml` que orquestre a aplicação e suas dependências (banco de dados). Um `Makefile` para automatizar os passos de build e execução é fortemente recomendado.

O Que Não é o Foco

Para que você possa focar no que mais valorizamos, não é necessário se preocupar em:

- boilerplates complexos ou frameworks completos. Uma aplicação simples e funcional é o ideal.
- Implementar uma interface web (API) completa com múltiplos endpoints, tratamento de autenticação, etc. Uma CLI bem documentada ou um único endpoint HTTP é suficiente.
- Automatizar o download dos arquivos de dados.

Aguardamos sua solução. Boa sorte!