code-invest-c4-model

Code Invest

Visão Geral

Este repositório do GitHub contém um projeto de simulação de uma corretora de valores, com funcionalidades completas de compra e venda de ações através do home broker, visualização de indicadores financeiros via gráfico, além de outras características.

Objetivo

O objetivo deste projeto é fornecer uma plataforma simulada de corretora de valores que permita aos usuários realizar operações de compra e venda de ações, acompanhar os indicadores financeiros em tempo real através de gráficos e receber notificações sobre mudanças significativas no mercado de ações. O projeto é construído usando tecnologias modernas como Go, Next.js, Nest.js, React e Apache Kafka.

Principais Características

1. **Home Broker**: O projeto oferece uma interface intuitiva de home broker, permitindo que os usuários realizem operações de compra e venda de ações de forma rápida e fácil. Eles podem visualizar informações detalhadas sobre as ações, incluindo preços, volume de negociação e tendências históricas.

2. **Visualização de Indicadores**: O sistema utiliza gráficos interativos para exibir os principais indicadores financeiros, como preço das ações ao longo do tempo, volume de negociação, médias móveis, índices de mercado, entre outros. Os usuários podem personalizar a exibição dos gráficos e explorar diferentes períodos de tempo.

3. **Notificações de Mercado**: Os usuários têm a opção de receber notificações em tempo real sobre eventos relevantes no mercado de ações. Isso pode incluir mudanças significativas nos preços das ações, anúncios de empresas, divulgação de resultados financeiros, entre outros. As notificações são entregues através de diferentes canais, como e-mail, SMS ou notificações push.

4. **Integração com Apache Kafka**: O projeto utiliza o Apache Kafka como sistema de mensagens para processar e transmitir dados em tempo real. Ele permite que as informações de mercado sejam atualizadas continuamente e distribuídas para os usuários de forma eficiente e escalável.

Tecnologias Utilizadas

O projeto faz uso das seguintes tecnologias:

- **Go**: É a linguagem de programação principal utilizada para desenvolver a lógica de negócios do projeto. Go é conhecida por sua eficiência, simplicidade e suporte para concorrência, tornando-a uma escolha adequada para aplicações de alto desempenho e escaláveis.
- **Next.js**: É um framework de desenvolvimento web em React que permite a criação de aplicações SSR (Server-Side Rendering) e SSG (Static Site Generation). Ele é usado para construir a interface do usuário da aplicação, fornecendo uma experiência interativa e responsiva.
- **Nest.js**: É um framework back-end em Node.js que facilita a criação de aplicativos escaláveis e eficientes. Ele fornece recursos como injeção de dependência, arquitetura em camadas e uma sintaxe declarativa para simplificar o desenvolvimento do servidor de aplicação.
- **React**: É uma biblioteca JavaScript para construção de interfaces de usuário. O React é usado em conjunto com o Next.js para criar componentes reutilizáveis, gerenciar o estado da aplicação e criar uma experiência de usuário

dinâmica e responsiva.

- **Apache Kafka**: É uma plataforma de streaming distribuída que permite o processamento e a transmissão de eventos em tempo real. Ele é usado para a transmissão de dados do mercado de ações, atualizações de preços e notificações para os usuários do sistema.

Contribuição

Contribuições para este projeto são bem-vindas. Se você deseja contribuir, siga as práticas recomendadas para desenvolvimento, faça fork do repositório, crie uma branch específica para sua contribuição e envie um pull request quando estiver pronto.

Lembre-se de fornecer uma descrição clara das alterações que você está propondo e adicione testes adequados para garantir a qualidade do código.

Licença

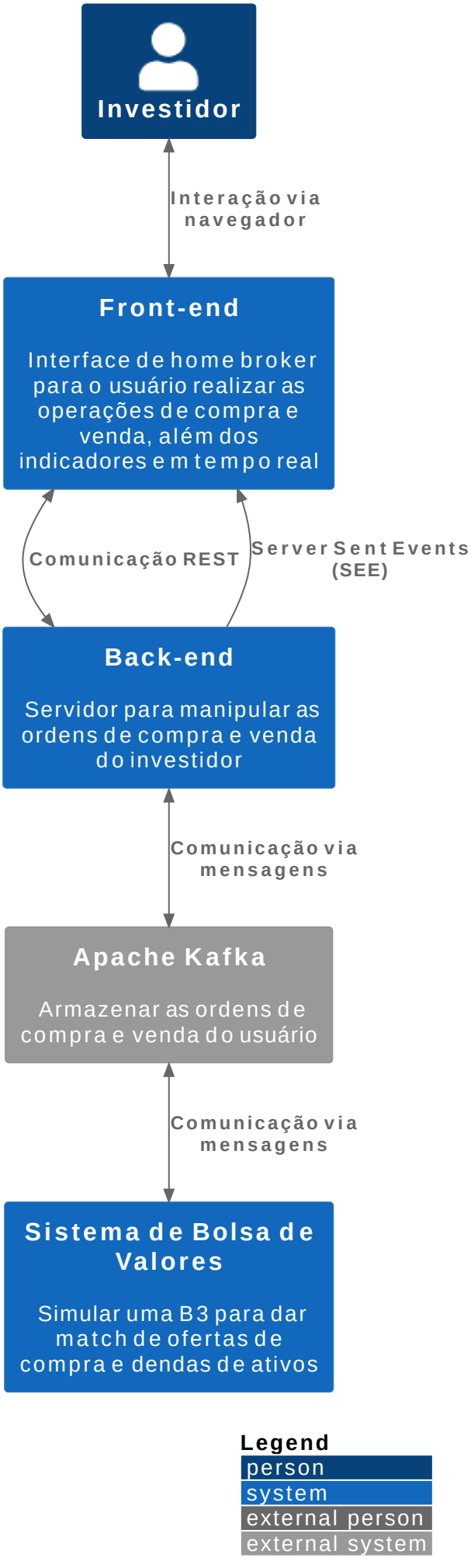
Este projeto é licenciado sob a MIT License, o que significa que você é livre para usá-lo, modificá-lo e distribuí-lo sob os termos da licença.

Aviso Legal

Este projeto é uma simulação e não representa uma corretora de valores real. As operações realizadas e os indicadores exibidos são apenas para fins ilustrativos. Não deve ser considerado como aconselhamento financeiro ou recomendação de investimento. Sempre consulte um profissional financeiro qualificado antes de tomar qualquer decisão de investimento.

c1-context

`/c1-context`


Investidor
Interação via navegador
Front-end
Interface de home broker para o usuário realizar as operações de compra e venda, além dos indicadores em tempo real
Comunicação REST
Server Sent Events (SEE)
Back-end
Servidor para manipular as ordens de compra e venda do investidor
Comunicação via mensagens
Apache Kafka
Armazenar as ordens de compra e venda do usuário
Comunicação via mensagens
Sistema de Bolsa de Valores
Simular uma B3 para dar match de ofertas de compra e dendas de ativos
Legend
person
system
external person
external system

A plataforma de simulação de corretora de valores possui uma arquitetura em camadas, onde o frontend é desenvolvido em React e Next.js, o backend em Nest.js, o sistema de bolsa de valores em Go, e o Apache Kafka é utilizado como intermediador entre o sistema de bolsa de valores e o backend, permitindo a comunicação assíncrona e em tempo real.

Frontend (React e Next.js): O frontend da plataforma é construído utilizando React, uma biblioteca JavaScript para construção de interfaces de usuário. O Next.js, um framework baseado em React, é utilizado para possibilitar a renderização do lado do servidor (SSR) e a geração de sites estáticos (SSG). Essa combinação permite criar uma experiência de usuário dinâmica e responsiva.

A interface do usuário é desenvolvida utilizando componentes reutilizáveis, gerenciando o estado da aplicação com React Hooks e consumindo dados do backend por meio de APIs RESTful ou GraphQL. A interação do usuário com a plataforma é feita por meio de páginas, formulários e elementos visuais que permitem a visualização das informações da bolsa de valores e a realização de operações de compra e venda de ações.

Backend (Nest.js): O backend da plataforma é desenvolvido utilizando o framework Nest.js, que é construído em cima do Node.js. O Nest.js fornece uma arquitetura modular baseada em componentes, com injeção de dependência e uma sintaxe declarativa que facilita o desenvolvimento e a manutenção do servidor de aplicação.

O backend é responsável por lidar com as requisições do frontend, processar a lógica de negócios e se comunicar com o sistema de bolsa de valores. Ele oferece APIs RESTful ou GraphQL para fornecer os dados necessários para o frontend e processar as solicitações de compra e venda de ações.

Sistema de Bolsa de Valores (Go): O sistema de bolsa de valores é implementado em Go, uma linguagem de programação conhecida por sua eficiência, simplicidade e suporte à concorrência. O sistema de bolsa de valores é responsável por gerenciar o fluxo de dados e informações relacionadas às ações, incluindo preços, volume de negociação, tendências históricas e outras métricas financeiras.

Ele se comunica com o backend através de uma interface definida, seja por meio de chamadas diretas a APIs ou por meio do Apache Kafka.

Apache Kafka: O Apache Kafka é um sistema de mensagens distribuído que atua como intermediador entre o sistema de bolsa de valores e o backend. Ele permite o processamento e a transmissão de eventos em tempo real, garantindo a entrega confiável e escalável das mensagens.

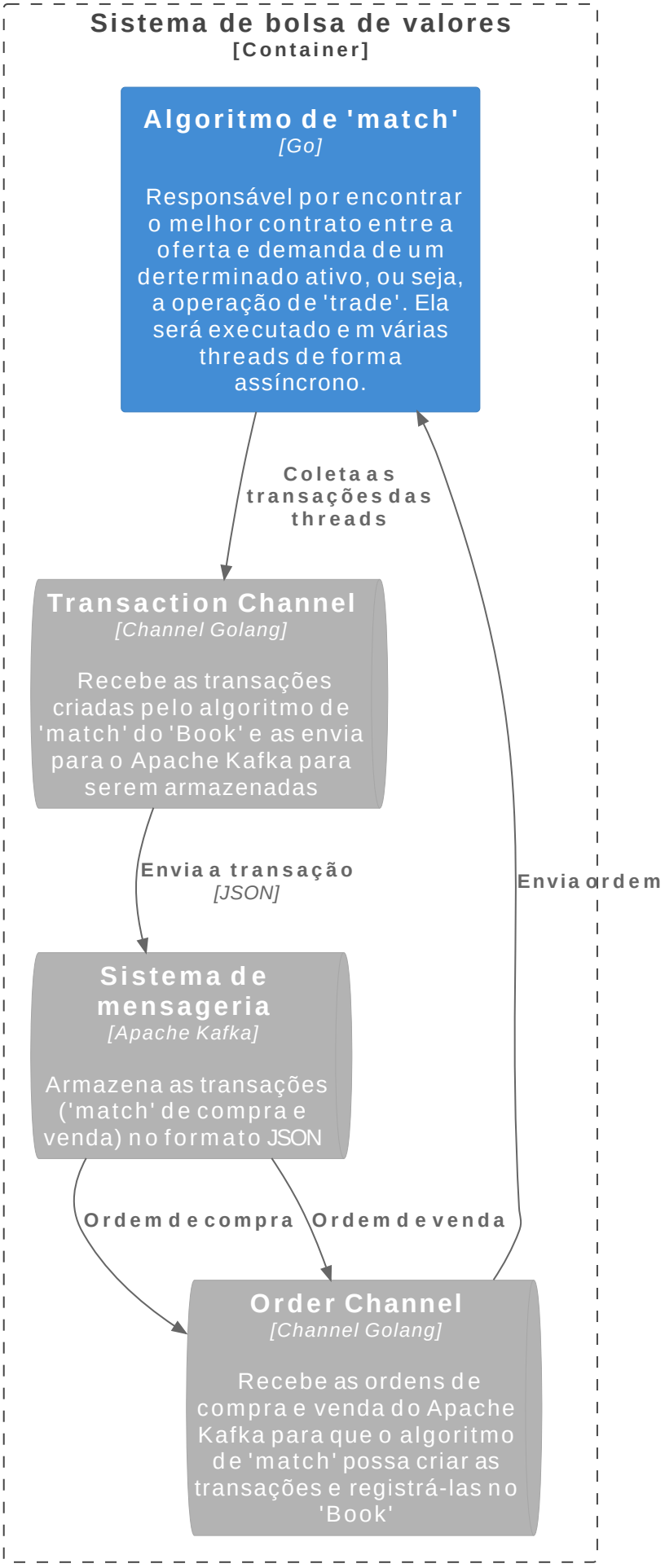
O sistema de bolsa de valores publica as atualizações de preços, volumes de negociação e outras informações relevantes no Kafka, enquanto o backend consome essas mensagens, atualizando as informações necessárias no banco de dados e fornecendo-as para o frontend. Essa abordagem assíncrona e em tempo real permite uma visualização precisa e atualizada das informações da bolsa de valores no frontend.

Em resumo, o frontend desenvolvido em React e Next.js fornece a interface do usuário, o backend em Nest.js lida com a lógica de negócios e se comunica

com o sistema de bolsa de valores em Go, e o Apache Kafka atua como intermediador para garantir a transmissão confiável e em tempo real dos dados entre o sistema de bolsa de valores e o backend.

c2-container

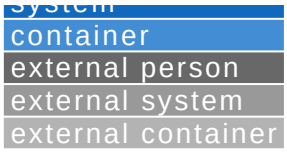
/c2-container



Legend

person

system



No segundo nível podemos verificar, de forma mais detalhada, como a arquitetura do *Code Invest* está interligada aos seus componentes, além das tecnologias que fazem parte deste ecossistema.

Requisitos funcionais

- Simulador da bolsa possui um algoritmo ligeiramente complexo para fazer o match das ordens de compra e venda

Requisitos não funcionais

- As operações devem ser "*in memory*" a fim aumentar a agilidade na execução do mesmo, e para isso as principais alocações precisam ficar na memória *heap*
- Garantia de *memory safe* a partir do recurso de *channels* da linguagem Go

Algoritmo de *match* das ordens de compra e venda

Esse algoritmo consiste em 2 filas (uma de compra e uma de venda) e será feito comparações entre elas para conseguir conciliar a oferta com a demanda da melhor forma possível. Veja a ilustração abaixo:

Cada vez que uma ordem de compra e venda dão "match", é gerado uma transação que será publicada no Apache Kafka no formato JSON.

Memory safe com *channels*

A fim de evitar erros como *race condition* o qual ocorre quando 2 ou + threads tentam alterar um valor ao mesmo tempo.

Para isso, como solução foi adotado o recurso de *channels* da linguagem Go que fornece um "canal" de comunicação entre 2 ou + threads e o dado que está nesse canal será coletado por uma dessas threads conectadas a esse canal.

O papel das *channels* na aplicação é para que todas as ordens de compra e venda sejam enviadas a um único *channel* de input para serem registradas no "livro" (*Book*) de compra/venda.

Além disso, quando ocorre um "match" é criado uma transação que será enviada a um *channel* de output para assim ser armazenado no Apache Kafka

c3-component

`/c3-component`