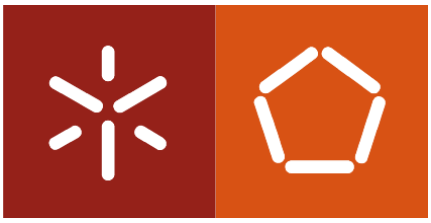

UNIVERSIDADE DO MINHO
Mestrado Integrado em Engenharia Informática

PARADIGMAS DE SISTEMAS DISTRIBUÍDOS

Exchange - Sistema de gestão de acções

Grupo 2
Alexandre Silva — a73674
André Cunha Santos — a61778
Renato Rebelo — pg33872

Braga, 18 de Fevereiro de 2018

Resumo

Este trabalho consiste na implementação de um sistema distribuído de compra e venda de acções de empresas. Neste sistema os utilizadores têm uma aplicação cliente que interage com um servidor e pode receber notificações de eventos sempre que uma compra/venda de interesse é efectuada. Este servidor recebe os pedidos dos clientes e redireciona-os para as exchanges adequadas. Utiliza-se ainda um directório, que disponibiliza uma API REST para consulta da localização das exchanges e ainda histórico das transações efectuadas.

Conteúdo

1. Arquitetura geral e componentes

Neste sistema existem 5 componentes: uma aplicação **cliente** escrita em java com a qual um utilizador pode interagir para fazer pedidos de compra e venda que são enviados para um **servidor** em Erlang. Este servidor em Erlang tem a função de encaminhar os pedidos do cliente para as **exchanges** corretas, consoante as empresas referenciadas nos pedidos à ao servidor. Existe ainda um **diretório** em Java que disponibiliza uma API REST, recorrendo à *framework* `Dropwizard`. Neste directório são guardados os endereços e portas das exchanges assim como o histórico das transações efetuadas. A informação do directório é usada pelo servidor para saber a localização das exchanges onde cada empresa é negociada e com isso poder redreccionar os pedidos. As exchanges por sua vez implementam a lógica de negócio do sistema e processam os pedidos de compra e venda de acções. Quando uma ordem é satisfeita, as exchanges notificam um **broker** feito em Java e informam o directório da transação. O broker por sua vez notifica os clientes das transações. A interação das exchanges com o broker e do broker com o cliente foi feita usando *ZeroMQ*. As exchanges possuem um socket PUB que comunica com um socket XSUB do broker Este socket por sua vez envia o que recebe para o socket XPUB do proxy e este por sua vez está ligado a sockets SUB dos clientes. A arquitectura geral da aplicação pode ser vista na figura 1.1.

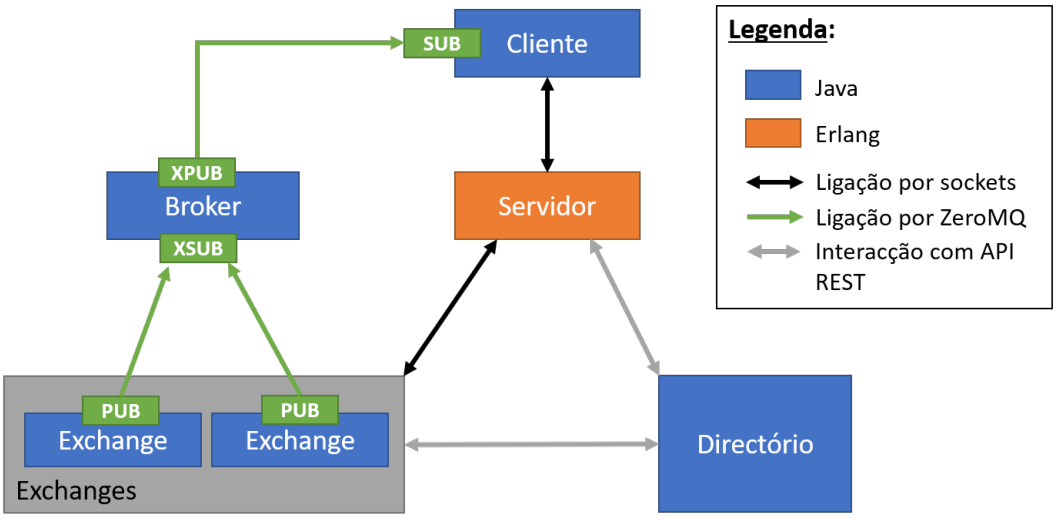

Figura 1.1: Estrutura geral da aplicação

Embora na figura 1.1 a título exemplificativo apenas se represente um cliente e duas exchanges, importa referir que esta arquitectura suporta vários clientes e mais que duas exchanges. A implementação de cada um dos componentes será discutida com mais detalhe nas secções seguintes.

2. Cliente

Através da aplicação cliente, um utilizador pode indicar comandos e enviar pedidos ao servidor. Os comandos são os seguintes:

- *sub* <empresa> - O utilizador indica a intenção de subscrever as transações de uma empresa.
- *unsub* <empresa> - O utilizador indica a intenção de cancelar a subscrição a uma empresa.
- *login* <username> <password> - O utilizador faz login com um username e password.
- *logout* - O utilizador faz logout.
- *sell* <empresa> <n_acoes> <preco_min> - O utilizador indica a intenção de vender um determinado número de acções numa determinada empresa por um preço mínimo.
- *buy* <empresa> <n_acoes> <preco_max> - O utilizador indica a intenção de comprar um determinado número de acções numa determinada empresa por um preço máximo.
- *historico* <empresa> - Permite consultar o histórico de transações de uma empresa.

O cliente guarda informação de sessão sobre o utilizador, nomeadamente se este tem login feito e com que username. Algumas acções apenas são permitidas com login efetuado. O cliente guarda também informação de estado das subscrições ativas, não deixando o cliente subscrever mais que 10 empresas.

Os pedidos de login, logout, sell e buy são transformados em mensagens *protobuf* e enviados ao servidor. Após este envio, o cliente fica à espera de resposta do servidor. Os pedidos de PUB/SUB geram uma mensagem ZeroMQ ao broker. Os pedidos relativos ao histórico são feitos consultando o directório através de um pedido à API REST.

3. Servidor

O servidor foi implementado em Erlang e tem como objectivo receber pedidos do cliente sob a forma de mensagens protobuf e redireccionar para a exchange correcta. Para saber a localização das exchanges, o servidor contacta o directório ao iniciar, através de um pedido à API REST.

O servidor ao receber um pedido faz algumas verificações, por exemplo, mantém informação sobre que utilizadores possuem login efetuado e não permite operações de compra e venda de ações sem login efectuado. Uma vez que mantém uma lista de empresas (obtidas do directório), o servidor verifica também se as empresas indicadas pelo utilizador existem de facto. Em todos estes casos, é enviada uma resposta ao cliente, também através de mensagens *protobuf*. Os utilizadores que o servidor conhece são inicializados no arranque com valores predefinidos.

Se tudo o utilizador tiver indicado uma ordem de compra, tiver login efectuado e a empresa for válida, é enviado um pedido de compra para a exchange onde a empresa está a ser negociada, também usando mensagens *protobuf*.

Quando um cliente se conecta ao servidor é criado um novo processo para tratar dos pedidos desse cliente. Uma vez que os processos erlang são leves, isto permite que muitos clientes se possam ligar ao servidor. Ao enviar os pedidos às exchange, o servidor cria uma conexão por pedido.

4. Directório

O directório tem a função de manter uma "base de dados" de empresas e exchanges em que estas são negociadas. Além disso mantém ainda registos das transações efectuadas nas exchanges. Este componente foi implementado usando a framework `Dropwizard` em Java que permite a consulta desta informação através duma API REST.

Esta API é usada pelo servidor, para obter a lista de empresas e as exchanges em que estas são negociadas através de pedidos GET e assim poder redirecionar os pedidos que recebe dos clientes. É também usada pelas exchanges, que através de pedidos POST colocam informação sobre o seu endereço/porta através da qual aceitam conexões. Além disso as exchanges através de pedidos POST podem adicionar ao directório informações de transações completadas para efeitos de manutenção de um histórico de transações.

O directório tem os seguintes endpoints:

- `/empresas` - Obtém lista de empresas e respectiva informação. A informação inclui o endereço e porta da exchange onde as acções de cada empresa são negociadas. Aceita pedidos GET, POST e DELETE. O pedido POST recebe como parâmetros as informações da empresa e o pedido DELETE recebe como parâmetros o nome da empresa.
- `/historico` - Obtém histórico global de transações. Aceita pedidos GET e POST. O pedido POST aceita como parâmetros as informações de uma transação completada, tais como entre que utilizadores ocorreu a transação, em que empresa, quando e por que preço.
- `/historico/<empresa>` - Obtém histórico de transações de uma empresa. Aceita pedidos GET.

5. Exchanges

As *exchanges* estão implementadas em Java e implementam a lógica de negócio do sistema. Ao serem inciadas, as *exchanges* enviam informação ao directório sobre as empresas cujas acções nela são negociadas e o seu endereço. O nome das empresas de uma *exchange* podem ser passadas como argumento ao programa.

Depois disso, as *exchanges* recebem pedidos de ordens do servidor em mensagens *protobuf* e tentam satisfazer a ordem de compra/venda quando ela chega. Quando uma ordem não pode ser satisfeita, é guardada numa lista para que possa ser satisfeita mais tarde. Quando uma ordem é satisfeita, é enviada uma mensagem ZeroMQ ao broker, para que este possa notificar os clientes interessados. É ainda enviado um pedido POST ao directório para que a transação seja registada no histórico.

As exchanges são multi-threaded e criam uma nova thread para tratar cada conexão nova que recebem. Uma vez que o servidor cria uma conexão à exchange por pedido, na prática tem-se uma thread por pedido nas exchanges. Por se tratar de um programa multi-threaded, houve cuidado de fazer controlo e concorrência nas classes relevantes.

6. Broker

Foi criado um broker também em Java, que recebe as notificações das exchanges e as entrega aos clientes interessados.

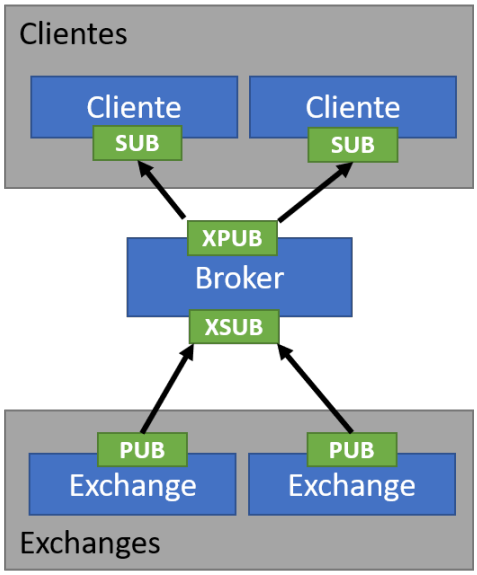

Figura 6.1: Broker usando sockets ZeroMQ XPUB e XSUB

7. Conclusão

Os objetivos propostos para o trabalho foram cumpridos. Foi implementado um sistema que permite a compra e venda de ações de empresas de acordo com as especificações pretendidas. Neste trabalho foram usadas todas as tecnologias, linguagens e ferramentas lecionadas ao longo do semestre, nomeadamente Erlang, Java, Dropwizard, ZeroMQ e protocol buffers.

Como futuras melhorias sugere-se a implementação de algumas funcionalidades extra e melhoria de algumas funcionalidades existentes. Seria interessante por exemplo o uso de bases de dados para tornar os dados de alguns componentes persistentes, como por exemplo os utilizadores do servidor e a informação nas exchanges. Apesar de no código erlang do servidor ser possível registar novos utilizadores, essa funcionalidade não é fornecida aos clientes, no entanto seria possível faze-lo com alguma facilidade. A comunicação das exchanges com o broker e deste com os clientes é feita através de mensagens ZeroMQ que possuem strings. Em vez de strings, uma possível melhoria seria enviar mensagens protobuf com ZeroMQ por forma a estruturar melhor os dados das notificações.