Universidade do Minho

Mestrado Integrado em Engenharia Informática

Paradigmas de Sistemas Distribuídos

PeerLending: serviço de empréstimos entre pares

Grupo 13

Autores:

Bernardo Mota	(A77607)
Filipe Nunes	(A78074)
Luís Neto	(A77763)

20 de Janeiro de 2019

1 Introdução

O presente documento tem como finalidade justificar as escolhas do grupo na realização deste projeto. Este projeto foi proposto na unidade curricular de Paradigmas de Sistemas Distribuídos e consiste na implementação de um serviço de intermediação de empréstimos a empresas por parte de investidores.

Atualmente, este serviço pode ser visto com um certo grau de importância, pois as empresas recorrem, cada vez mais, a empréstimos feitos por investidores.

Ao longo do presente documento será apresentado de forma mais detalhada cada um dos componentes do serviço e será descrita a sua função, tecnologias utilizadas e a justificação das mesmas.

2 Descrição do problema

O serviço deverá ser constituído por 4 componentes fundamentais: Cliente, Front-End, Exchange e Diretório.

O serviço deve então ser capaz de autenticar os clientes, receber os pedidos feitos por eles e encaminhar os pedidos para a Exchange correta. Os Clientes devem poder realizar vários pedidos e subscrever-se para obter notificações acerca de leilões e emissões que estejam a decorrer e que sejam do seu interesse.

Cada Exchange, por sua vez, estará encarregue de processar os pedidos feitos pelo cliente às empresas, que estão previamente atribuídas a cada Exchange. Para além disso, estas também são responsáveis pelo envio das notificações para os Clientes interessados.

O sistema ainda deve disponibilizar uma interface RESTful, através do diretório. Este receber pedidos efetuados pelos clientes para visualizar, em tempo real, os leilões e as emissões que estão a decorrer no sistema.

3 Arquitetura do Sistema

A seguir será apresentada a arquitetura do sistema desenvolvido, assim como a descrição de cada componente do sistema e de algumas decisões de implementação.

3.1 Visão Geral

A seguir serão descritos os componentes que compõe o sistema *PeerLending*.

O **Cliente** (em Java) é a interface do utilizador com o programa, que comunica com a *Front-End* para fazer pedidos, recebendo a resposta posteriormente. O ***Front-End*** (em Erlang) recebe os pedidos do utilizador e encaminha-os para a *Exchange* correspondente, fazendo chegar a resposta desta ao Cliente (para além de fazer autenticação). A ***Exchange*** (em Java) processa os pedidos que o *Front-End* faz chegar, atualizando o Diretório, notificando os interessados através do *Broker* e enviando a resposta para a *Front-End*. O **Diretório** mantém a informação sobre as Empresas, Leilões e Emissões, e responde a pedidos *HTTP* sobre os dados (como uma consulta dos leilões por um cliente).

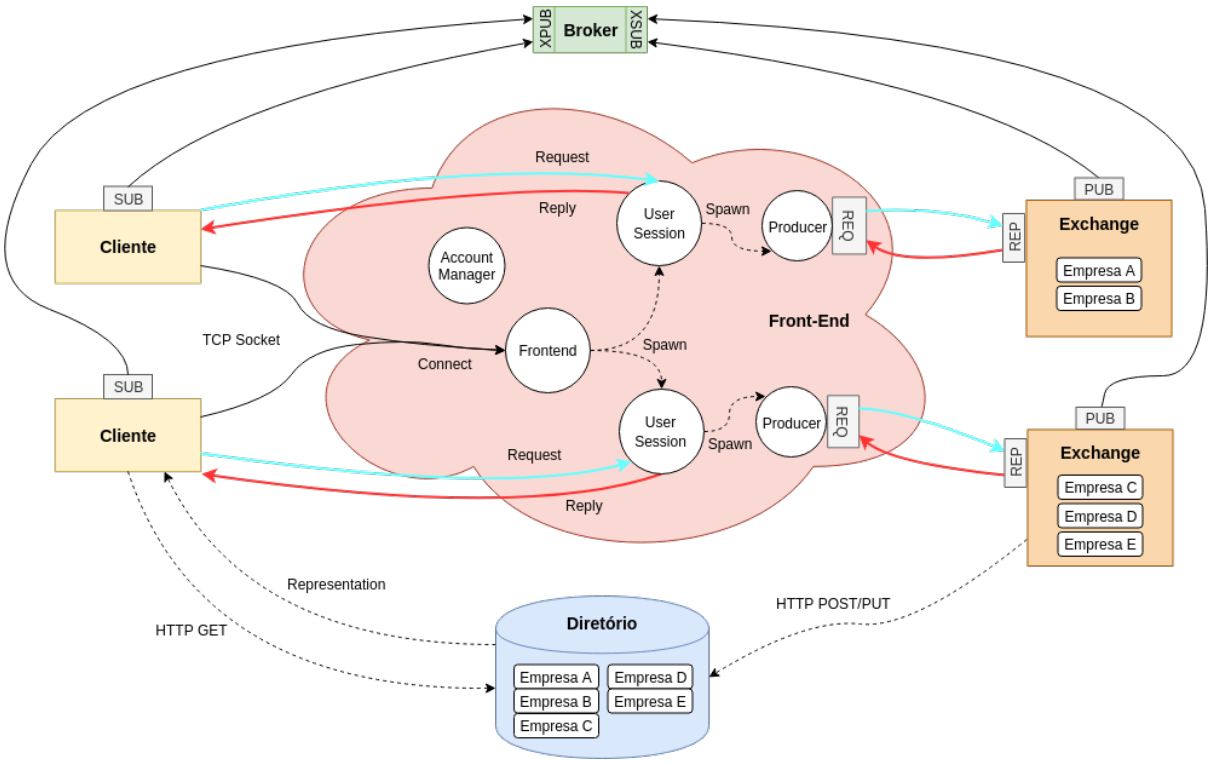

Figura 1: Visão geral do sistema

Como podemos verificar nesta maquete do sistema, o Cliente conecta-se à *Front-End* através de um socket TCP e, depois de feita a autenticação, comunica com um ator de *Erlang* que faz chegar os pedidos à Exchange correspondente. A comunicação deste ator

com a *Exchange* é feita através de um socket *ZeroMQ REQ*, que envia o pedido e recebe a resposta para enviar para o Cliente.

A *Exchange* recebe pedidos através do seu socket *REP*, processando o pedido, alterando os dados do sistema para refletir o pedido e enviando a resposta (se o pedido for um sucesso). Para além disto, atualiza o Diretório através de pedidos *HTTP POST* e *HTTP PUT*.

Podemos também verificar que existe um sistema de notificações, conseguido com sockets *SUB* e *PUB* do *ZeroMQ* e com um *Broker XPUB/XSUB*, que intermedeia as subscrições e notificações do sistema. As subscrições do Cliente são enviadas para todas as *Exchanges* e as notificações resultantes de ações do sistema são enviadas pelas *Exchanges* para todos os clientes interessados. Este padrão revelou-se ser o mais adequado, visto que as notificações precisam de ser em tempo real e o modelo de subscrição enquadra-se nas necessidades do sistema.

3.2 Cliente

O Cliente é o componente que permite a interação do utilizador, que pode aceder às funcionalidades do programa e introduzir dados para realizar pedidos, que são enviados para o sistema para serem realizados.

O componente foi feito em *Java*, utiliza sockets TCP para comunicar com a *Front-End* (através de um endereço conhecido) e utiliza sockets *ZeroMQ SUB* para subscrever e receber notificações do *Broker*.

Quando o componente é iniciado, é utilizada uma *thread* para tratar das notificações (subscrever e receber notificações em tempo real) e é iniciada uma conexão com o *Front-End*. O utilizador introduz as suas credenciais e é enviado um pedido para o servidor. Caso as credenciais sejam autenticadas, é apresentado o menu com as funcionalidades disponíveis, que dependem do tipo do utilizador (Empresa ou Investidor).

Após selecionar uma opção do menu e preencher os dados do pedido, este é enviado para o servidor que, posteriormente, envia a resposta do pedido, que consiste no sucesso deste.

A *thread* de notificações recebe todas as notificações sobre os tópicos subscritos (leilões e emissões de várias empresas) e informa o utilizador em tempo real das ações do sistema, como, por exemplo, uma nova licitação ou o término de um leilão.

3.3 Front-End

A implementação do *Front-End* em *Erlang* permite a utilização de processos *lightweight* isolados para construir a lógica de autenticação e sessão do utilizador de uma maneira simples e eficiente. Utilizando este paradigma conseguimos obter um sistema claro e eficiente, com delegação de papeis entre 4 tipos de atores.

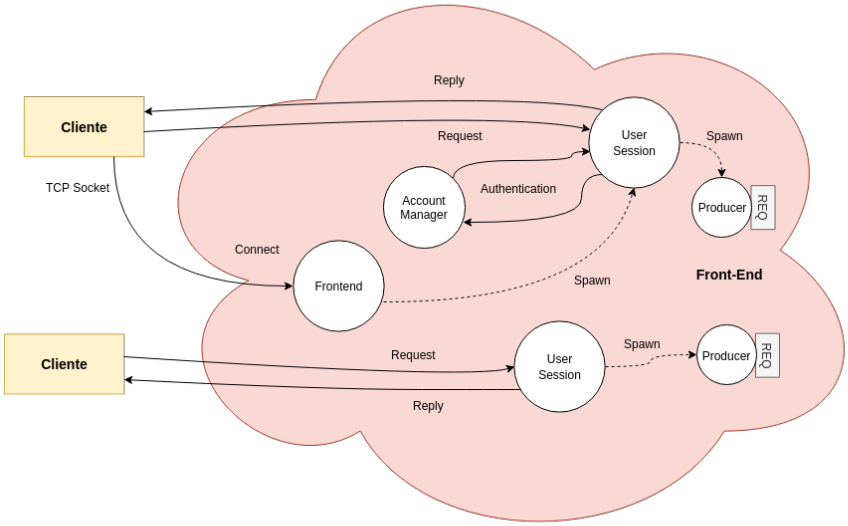

Figura 2: Maquete do *Front-End*

O ator *Frontend* recebe as conexões dos Clientes e cria um ator *UserSession* que trata da sessão do utilizador. O ator *UserSession* recebe as credenciais de autenticação do Cliente e envia-as para o ator *AccountManager* que, por sua vez, envia o sucesso da operação.

Depois de autenticado, o *UserSession* recebe os pedidos do Cliente e cria um ator *Producer* para tratar com a comunicação com a Exchange desse pedido, permitindo que a *UserSession* receba mais pedidos.

O *Producer* calcula a *Exchange* a que o pedido tem de ser enviado, através do nome da Empresa contido no pedido, e conecta-se ao endereço da *Exchange* com um socket *REQ*. O pedido é enviado e, após o pedido ser efetuado, a resposta é reencaminhada para o Cliente.

3.4 Exchange

A *Exchange*, feita em Java, é responsável por processar os pedidos de acordo com a lógica estabelecida e por efetuar as alterações aos dados consequentes.

Este componente recebe os pedidos do *Front-End* através de um socket *REP* e, dependendo do tipo de pedido, efetua várias ações, caso o pedido seja válido. Toda a lógica definida no enunciado sobre leilões, emissões a taxa fixa e licitações encontra-se definida aqui.

Se o pedido for válido (segundo as regras definidas no enunciado), são alterados os dados do sistema presentes na *Exchange*, é enviada uma notificação sobre a ação através do socket *PUB* para o *Broker* e é feito um pedido *HTTP POST* ou *PUT* para o Diretório para atualizar ou adicionar dados do sistema. Para além disto, é enviada a resposta para o *Front-End*, para ser redirecionada para o Cliente.

3.5 Diretório

O Diretório disponibiliza uma interface *REST* para obter a informação sobre os dados do sistema e para atualizar ou acrescentar dados. Desta forma, é disponibilizada uma interface uniforme sobre os vários recursos do sistema, que beneficia das vantagens da utilização do *HTTP*. É utilizada a framework *Dropwizard* para construir o serviço *RESTful*.

Estão disponíveis 4 recursos distintos: **empresa**, **leilao**, **emissao** e **licitacao**. Cada um tem operações *GET* e *POST* definidas, permitindo visualizar dados ou adicionar dados a todos os recursos, através dos URIs hierárquicos. Os recursos **leilao**, **emissao** e **licitacao** têm a operação *PUT* definida, para atualizar informação sobre o respetivo recurso.

A seguir apresentam-se exemplos de alguns pedidos *HTTP* possíveis e a descrição do seu resultado:

- ***GET /empresas***

  Retorna a representação de todas as empresas.

- ***GET /empresa/leilao?nome=x&idLeilao=0***

  Retorna a representação com informação de um leilão de uma determinada empresa.

- ***POST /empresa/leilao***

  Adiciona um leilão, cujos parâmetros são definidos no corpo da mensagem *HTTP* através de serialização para *JSON*.

- ***POST /empresa/emissao/licitacao***

  Adiciona uma licitação a uma emissão, cujos parâmetros são definidos no corpo da mensagem *HTTP*.

- ***PUT /empresa/leilao***

  Atualiza informação sobre um leilão, cujos parâmetros são definidos no corpo da mensagem *HTTP*. Este pedido é feito no fim de um leilão pela *Exchange* para atualizar o estado e sucesso do leilão.

O Diretório também envia o código de estado *HTTP* com a resposta, enviando o código de erro 404 caso não tenha encontrado a informação requerida ou enviando um código de sucesso caso o pedido tenha sido completo (200 *OK* no caso de *GET* ou 201 *Created* no caso de *POST* ou *PUT*).

3.6 Serialização

Para lidar com a heterogeneidade de linguagens de programação do sistema, foi necessária a utilização de Serialização para obtermos uma representação neutra dos dados transmitidos entre componentes de linguagens com representação de dados distintas.

Foi utilizado ***Protocol Buffers*** para fazer a serialização dos dados transmitidos entre o Cliente e o *Front-End* e entre o *Front-End* e a *Exchange*.

3.7 Notificações

Para receberem a informação, em tempo real, sobre os leilões e emissões a taxa fixa em que estão interessados, os utilizadores podem receber notificações, podendo escolher subscrever todos os leilões ou emissões, ou subscrever leilões ou emissões de empresas específicas.

Quando é criado um novo leilão ou emissão, ou quando há licitações em leilões ou emissões, a *Exchange* envia uma notificação com um formato específico, para ser encaminhada para todos os clientes interessados.

Para a disseminação das subscrições e notificações entre os vários Clientes e as várias *Exchanges*, foi utilizado um *Broker XPUB/XSUB* que reencaminha todas as notificações para os clientes que registaram interesse no tópico.

4 Conclusão

O desenvolvimento deste projeto permitiu, através da implementação de várias técnicas, reunir e consolidar os conceitos aprendidos ao longo do semestre nesta unidade curricular.

O sistema criado utiliza várias linguagens e *middlewares* diferentes o que permitiu escolher aquelas que melhor se encaixam para estruturar os componentes existentes e a comunicação entre estes. Por sua vez, a utilização do *ZeroMQ* e da serialização dos dados no sistema irá permitir a interoperabilidade entre os diferentes componentes, de modo a que todo o serviço de empréstimos funcione adequadamente.

Concluiu-se que o projeto foi bem sucedido, uma vez que se conseguiu implementar tudo aquilo que estava proposto.