

Paradigmas de Sistemas Distribuídos

Trabalho Prático

-

Mestrado em Engenharia Informática
Universidade do Minho
Relatório

Grupo

PG41080	João Ribeiro Imperadeiro
PG41081	José Alberto Martins Boticas
PG41091	Nelson José Dias Teixeira

19 de Janeiro de 2020

Conteúdo

1	Introdução	2
2	O sistema	3
2.1	Funcionamento	3
2.2	Implementação	3
2.2.1	Servidor	3
2.2.1.1	server.erl	3
2.2.1.2	login.erl	3
2.2.1.3	client.erl	3
2.2.1.4	importer.erl	3
2.2.1.5	producer.erl	3
2.2.1.6	negotiator.erl	4
2.2.2	Cliente	4
3	Conclusão	5
4	Webgrafia	6

Capítulo 1

Introdução

Neste trabalho prático é requerido o desenvolvimento de um protótipo de uma plataforma de negociação entre fabricantes e importadores de produtos. Este protótipo é composto por cliente, servidor de *front-end*, negociador e catálogo, de entidades e negociações em curso. Os clientes podem existir em elevado número, sendo que cada um deles desempenhará sempre ou o papel de fabricante ou de importador. Cada fabricante indicará a disponibilidade para produzir um determinado artigo, numa quantidade mínima e máxima, a um preço mínimo (unitário), bem como o período de tempo durante o qual os importadores poderão fazer ofertas de encomenda (período de negociação). Por sua vez, cada importador indica a quantidade e valor unitário a que está disposto a pagar por um determinado artigo de um fabricante. Os clientes autenticam-se no servidor de *front-end*, o qual encaminha as suas ordens para um (de entre vários) negociadores. O catálogo disponibilizará uma interface *RESTful*, que permitirá obter informação sobre os fabricantes, importadores, e negociações em curso. Como tal, por forma a implementar este protótipo, foi utilizada a linguagem de programação *Java* (cliente, negociador e catálogo), *Erlang* (servidor *front-end*), e, ainda, *Protocol Buffers*, *ZeroMQ* e *Dropwizard*.

Capítulo 2

O sistema

2.1 Funcionamento

2.2 Implementação

2.2.1 Servidor

A implementação do servidor foi feita em *Erlang*. O servidor divide-se em diversos ficheiros, cada um dos quais corresponde a um tipo de ator, que se dividem em seis tipos:

- `server.erl`;
- `login.erl`;
- `client.erl`;
- `importer.erl`;
- `producer.erl`;
- `negotiator.erl`.

2.2.1.1 `server.erl`

Este é o primeiro ator criado, responsável pela criação de todos os outros atores e por aceitar novas conexões (clientes).

2.2.1.2 `login.erl`

Este tipo de ator é registado, como `loginHandler`, e criado pelo anterior. É o responsável pela autenticação e registo dos clientes, guardando todas as informações nesse sentido.

2.2.1.3 `client.erl`

Ator criado a cada nova conexão. Espera a receção de uma comunicação *TCP* com a autenticação do cliente e usando o `loginHandler` para confirmar a sua identificação. Segue-se a criação de novo ator, importador ou fabricante, que fica responsável pela comunicação com esse cliente. O ator atual é substituído pelo novo.

2.2.1.4 `importer.erl`

Ator criado pelo anterior, sempre que um cliente, do tipo importador, se autentica com sucesso. Fica então responsável pela comunicação com o importador, recebendo as suas encomendas e fazendo com que as mesmas cheguem ao respetivo negociador, através do `negotiatorsHandler`.

2.2.1.5 `producer.erl`

Semelhante ao anterior, mas para um cliente do tipo fabricante.

2.2.1.6 negotiator.erl

Existirá um ator deste tipo para cada negociador (quantidade pré-definida).

2.2.2 Cliente

Tal como pedido nos requisitos do trabalho, a implementação do cliente foi feita em *Java*. Para além de todas as classes nativas, foram ainda usadas funcionalidades das bibliotecas *ZeroMQ*, uma implementação em *Java* da ferramenta *ZeroMQ* para envio/receção de mensagens assíncronas, e *Protocol Buffers*, um formato de serialização de dados desenvolvido pela *Google*. Posto isto, passaremos ao detalhe do funcionamento do cliente.

O cliente, aquando do início da sua utilização por parte de um utilizador, começa por pedir os dados de autenticação e entra em contacto com o servidor para verificar a sua validade. Em caso afirmativo, é criada uma *thread* adicional que recebe informações do servidor e o utilizador é enviado para um de dois menus: o de fornecedor ou de importador. Se o utilizador for um importador, é lançada ainda outra *thread*, responsável por receber updates do catálogo quanto a produtores que o utilizador subscreveu.

De seguida, após toda esta configuração inicial, é mostrado um dos seguintes menus com as operações indicadas:

1. Importador:

- Oferta de encomenda;
- Subscrever notificações;
- Cancelar notificações.
- Atualizações.

2. Fornecedor:

- Oferta de produção;
- Atualizações.

Começando pelas operações do importador, temos que este pode encomendar um produto, tendo para isso de indicar o nome do produtor/produto, o número de unidades e o preço que está disposto a pagar por unidade. Pode ainda subscrever ou cancelar a subscrição de atualizações sobre um determinado produtor, bastando indicar o nome desse produtor.

Passando ao fornecedor, este pode apenas colocar uma oferta de produção, indicando o nome do produto, quantidade mínima/máxima, preço mínimo por unidade e período de negociação (em segundos).

Quanto às mensagens do servidor e do catálogo, estas são apresentadas sempre que há informações para mostrar, tendo em conta que o utilizador não está a realizar nenhuma operação ou quando este clica na opção "Atualizações".

Em ambos os casos, é dada a opção ao utilizador para sair da interface.

Capítulo 3

Conclusão

Capítulo 4

Webgrafia

- *Protocol Buffers:*
<https://developers.google.com/protocol-buffers>
- *Protocol Buffers - Java:*
<https://developers.google.com/protocol-buffers/docs/reference/java-generated>
- *Documentação - Java:*
<https://docs.oracle.com/en/java/javase/11/docs/api/index.html>