

# Sistema de negociação

Hugo Abreu | A76203      João Padrão | A76438  
João Reis | A75372

Paradigmas de Sistemas Distribuídos  
**Universidade do Minho**



23 de Janeiro de 2018

## 1 Introdução

O seguinte trabalho prático advém da necessidade de aplicação e consequente cimentação de conhecimentos adquiridos nas aulas relativas à unidade curricular de Paradigmas de Sistemas Distribuídos, visando temas como a correcta implementação de protocolos de comunicação inter-linguagens e compreensão das temáticas associadas. Assim sendo, o presente relatório reporta à fase final do projeto, estando, por conseguinte, visada a correta implementação de *Protocol Buffers*, tanto em Java com em Erlang. Nesta fase final é também esperado que os conhecimentos acerca de atores, implementação de abordagens com recurso a Message Oriented Middleware e serviços REST sejam corretamente aplicados.

Estratégias para a manipulação das tecnologias contempladas pela Exchange terão de ser adotadas. No final deste projeto, espera-se que tenham sido corretamente aplicados conceitos com significativa complexidade no âmbito da disciplina, de um modo organizado e estruturado, a fim de lançar as fundações adequadas à execução deste último segmento da componente prática de avaliação da unidade curricular.

## 2 Front-end Server

## 3 Exchange Server

**\*\*FALAR DO TIMER PARA O CLOSE\*\***

## 4 Directory Server

Para projetar o servidor de directório, ou seja, o servidor que contém todos os meta-dados relativos ao sistema de negociação, portanto, esta entidade do sistema mantém dados como, que empresas existem, em que exchange são transacionadas, que

exchanges existem e que empresas têm e, os diferentes preços diários das empresas em questão. A utilização de um servidor REST para este tipo de acesso a dados, é o mais correto, visto que não existe a necessidade de manter estado entre pedidos e, desta forma, faz com que um só servidor consiga suportar muitos utilizadores em simultâneo.

O servidor construído, tal como requerido pelo docente no enunciado, é um servidor REST, stateless, programado em Java com recursos a framework Dropwizard. Para ser possível garantir o acesso a todos os recursos que o serviço oferece, foi necessário definir *end-points* para obter e/ou atualizar esses mesmos recursos. Os *end-points* definidos para o servidor REST foram os seguintes:

- GET

- **/companies**

- Este *end-point* devolve uma lista com todas as empresas existentes no sistema. A lista devolvida é uma lista de objetos e, portanto, contém todos os atributos intrínsecos a cada empresa na enumeração.

- **/company/id**

- Este recurso, quando fornecido um correcto id de empresa, a respetiva empresa, caso esta se encontre efetivamente no sistema. Neste recurso é devolvida a empresa e todos os atributos a esta relacionados.

- **/company/id/today**

- O *end-point* descrito neste ponto, devolve a informação de preços do dia corrente, sendo que, esta informação também é devolvida no recurso anterior, desta forma, caso apenas seja necessário obter os preços, é possível reduzir a quantidade de informação que circula na rede.

- **/company/id/yesterday**

- Da mesma forma que o recurso mencionado anteriormente retorna os preços do dia atual, este recurso retorna os preços do dia anterior.

- **/exchanges**

- Este *end-point* devolve uma lista com todas as exchanges existentes no sistema. A lista devolvida é, tal como nas empresas, uma lista de objetos e, portanto, contém todos os atributos intrínsecos a cada exchange na enumeração.

- **/exchange/id**

- Quando o recurso descrito é acedido, ao fornecer um id de exchange válido, este recurso devolve a dita exchange com todos os atributos a ela relacionados, como por exemplo, o IP+Porta.

- **/exchange/id/companies**

- Este *end-point* devolve uma lista com todas as empresas da exchange que é fornecida como argumento (id). A lista devolvida é uma lista de objetos e, portanto, contém todos os atributos intrínsecos a cada empresa.

- PUT

Os recursos definidos como PUT, deverão ser idempotentes, ou seja, por muitas vezes que o recurso seja executado, o resultado é o mesmo e, portanto, é a operação ideal para, neste caso, o recurso listado abaixo.

– `/company/id/today`

Este recurso tem como objetivo atualizar os preços atuais de uma dada empresa e, portanto é uma operação idempotente visto que, o resultado é sempre o mesmo não importando a quantidade de vezes que é executado.

Todos os recursos abordados acima foram corretamente implementados e a sua execução foi efetivamente testada, o grupo teve o cuidado de utilizar as primitivas da framework Jackson para permitir a correcta serialização dos dados.

Foi também adicionada a funcionalidade ao REST server de às 23:59:59 de cada dia, executar a alteração diária, ou seja, às horas definidas, é accionado um *trigger* que vai a todas as empresas trocar os preços actuais para o dia anterior, garantindo assim a atualização das informações no directório.

## 5 Client

Para permitir o acesso às funcionalidades completas da Exchange e todos os recursos do eco-sistema associado, foi construído um Client que visa a utilização por utilizadores ditos normais.

No cliente foram aplicadas todas as funcionalidades requeridas pelo docente no enunciado disponibilizado, as opções de um utilizador com o login efetuado são as seguintes:

- [1] Ver empresas.
- [2] Ver exchanges.
- [3] Ver uma empresa específica.
- [4] Verificar preços da empresa.
- [5] Obter real time updates.
- [6] Cancelar real time updates.
- [7] Abrir posição.

As operações descritas são operações que, além de requeridas no enunciado do trabalho prático, são necessárias para a correta funcionalidade de toda a aplicação. A opção [1], contacta o serviço de directório (REST) de forma a obter a lista de todas as empresas que se encontram disponíveis para transacionar,

## 6 Conclusão

Com o término da fase final do projeto, é admissível afirmar que todos os objetivos requeridos pelo docente no enunciado cedido foram alcançados.

Conhecimentos relativos à implementação de Message Oriented Middleware, serviços REST, Protocol Buffers e atores em Erlang foram incrementados e por consequência cimentados. Em relação à linguagem Erlang, o grupo teve a necessidade de utilizar um gerenciador de pacotes semelhante ao Maven, o *rebar3* que, permitiu aprofundar o conhecimento relativo à linguagem e respetiva utilização de diferentes pacotes desenvolvidos nesta.

O maior foco incidiu sobre a correta implementação dos diferentes protocolos inter-linguagem e, em particular na implementação do front-end server que é parte fundamental na comunicação de sistemas.

A qualidade do trabalho apresentado agrada ao grupo, tendo havido tempo suficiente para o estudo das soluções encontradas ao problema proposto, sem precipitações e com espaço de maneio suficiente para que todos os membros pudessem inteirar-se não somente do segmento que lhes foi atribuído, mas também das restantes duas terças-partes que não exigiram tanta da sua atenção.