

Sistema de negociação

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



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

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.

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 recuso 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 descartando a quantidade de vezes que é executado. Este recurso também se encontra responsável por verificar a necessidade de trocar os valores, ou seja, verificar se já é um dia diferente e portanto tocar os valores para os dias corretos.

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.

5 Client

6 Conclusão