



# Mini-Komparator

## **PROJETO DE SISTEMAS DISTRIBUÍDOS (SD) ÉPOCA ESPECIAL**

DATA DE PUBLICAÇÃO: 2017-07-14

ÚLTIMA REVISÃO: 2017-07-14

Este documento descreve o cenário e os objetivos do projeto de época especial da unidade curricular de Sistemas Distribuídos.

# 1 CENÁRIO

O **Komparator** é um serviço de mediação de venda de produtos que faz uso de diversos fornecedores para satisfazer os pedidos dos seus clientes. Cada fornecedor tem um repositório de produtos, disponibilizando um serviço que permite efetuar pesquisas e compras de produtos.

A Figura 1 apresenta os componentes do sistema: os clientes, o mediador e os fornecedores; e também o registo de serviços.

Esta versão, Mini-Komparator, está focada apenas no serviço de fornecedor.

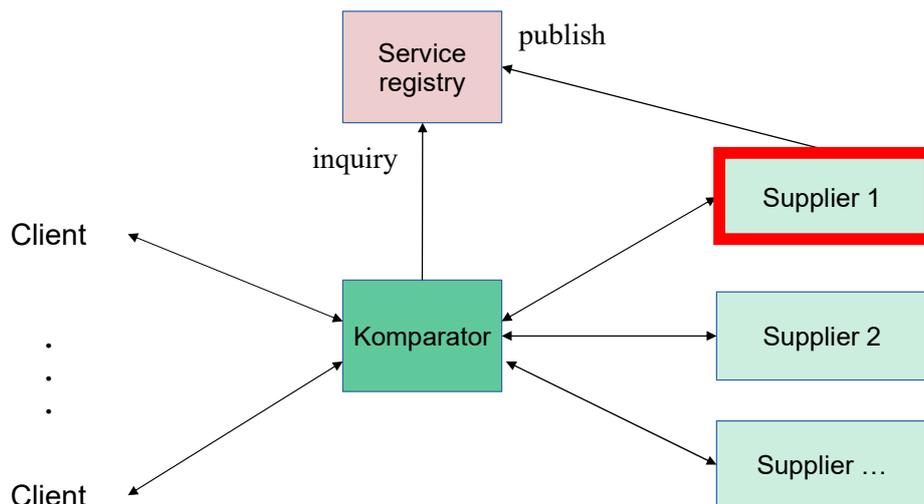


Figura 1 – Arquitetura do sistema.

## 2 PLATAFORMA DE DESENVOLVIMENTO

Todos os componentes do projeto serão implementados na plataforma Java, usando o JDK<sup>1</sup>. A construção de projetos deverá utilizar a ferramenta Maven.

A invocação remota de operações deve ser suportada por Web Services construídos com JAX-WS<sup>2</sup>. Para cada Web Service existirá um módulo servidor (ws) e um módulo cliente (ws-cli). Espera-se que cada módulo cliente (ws-cli) contenha *testes de integração*<sup>3</sup> que permitam verificar os requisitos do respetivo serviço (ws), quer nos casos normais, quer nos casos de erro. Os testes deverão ser construídos com a biblioteca JUnit.

Os serviços devem ser descobertos e localizados dinamicamente, por intermédio de um servidor UDDI<sup>4</sup>. Cabe a cada Web Service gerir o seu registo que associa o seu nome a um endereço.

Os serviços suportam o tratamento simultâneo de múltiplos pedidos (a cada pedido é atribuída uma tarefa de execução) e, portanto, será necessário sincronizar o acesso a variáveis partilhadas.

<sup>1</sup> Java Developer Kit – máquina virtual Java e respectivas ferramentas de desenvolvimento

<sup>2</sup> Java API for XML Web Services – implementação de Web Services SOAP em Java

<sup>3</sup> Integration Test – IT – de acordo com a convenção maven, estes testes executam-se na fase mvn verify

<sup>4</sup> Universal Description, Discovery, and Integration; <http://uddi.xml.org/>

## 3 REQUISITOS

---

O projeto está organizado em três partes que são descritas de seguida.

### 3.1 FORNECEDOR

O serviço de fornecedor (supplier-ws) disponibiliza operações de acesso ao catálogo de produtos. E também operações de compra. Este serviço deverá ser completado e testado.

As operações estão descritas no seguinte contrato WSDL<sup>5</sup>:

[http://disciplinas.tecnico.ulisboa.pt/leic-sod/2016-2017/labs/proj/supplier.1\\_0.wsdl](http://disciplinas.tecnico.ulisboa.pt/leic-sod/2016-2017/labs/proj/supplier.1_0.wsdl)

O ponto de partida para o projeto e os testes das operações principais estão disponíveis em:

[http://disciplinas.tecnico.ulisboa.pt/leic-sod/2016-2017/labs/proj/P1\\_start.zip](http://disciplinas.tecnico.ulisboa.pt/leic-sod/2016-2017/labs/proj/P1_start.zip)

[http://disciplinas.tecnico.ulisboa.pt/leic-sod/2016-2017/labs/proj/P1\\_eval.zip](http://disciplinas.tecnico.ulisboa.pt/leic-sod/2016-2017/labs/proj/P1_eval.zip)

#### 3.1.1 Operações auxiliares

Estas operações – *ping*, *createProduct*, *listProducts*, *listPurchases* - destinam-se a auxiliar os testes e não necessitam de ser testadas.

#### 3.1.2 Operações principais

A operação *getProduct* recebe um identificador de produto e devolve os dados desse produto. Cada produto contém o seu identificador, a descrição, a quantidade disponível e o preço praticado pelo fornecedor. Caso o produto não exista, é devolvido nulo.

A operação *searchProducts* recebe um texto de pesquisa (não pode ser vazio nem espaços) e devolve os produtos cuja descrição contenha o texto recebido. Cada produto contém o seu identificador, a descrição, a quantidade e o preço. A lista de resultados não é ordenada. Caso não sejam encontrados produtos, é devolvida uma lista vazia (mas não nula).

A operação *buyProduct* recebe um identificador e uma quantidade pretendida. Caso exista a quantidade pretendida, a compra é efetuada com sucesso e o inventário é atualizado. Como resultado é devolvido um identificador de compra gerado no momento da mesma.

#### 3.1.3 Novas operações principais

Estas operações deverão ser acrescentadas ao WSDL e depois implementadas e testadas.

A operação *setProductPrice* recebe um identificador de produto e um preço, e altera o preço desse produto. A operação *getProductPrice* recebe um identificador de produto e devolve o seu preço.

### 3.2 SEGURANÇA

A integridade das mensagens deve ser garantida com um MAC (*Message Authentication Code*).

Pode assumir que o cliente e o servidor partilham uma chave simétrica K, previamente distribuída.

A frescura das mensagens trocadas deve também ser garantida.

---

<sup>5</sup> Web Services Description Language – documento que descreve as operações de um serviço, respetivos argumentos, resultados e erros

### 3.3 TOLERÂNCIA A FALTAS

A tolerância a faltas é muito importante. Por essa razão pretende-se estender o serviço de fornecedor com suporte a replicação, para o dotar de uma maior disponibilidade.

Pretende-se seguir a abordagem de **replicação ativa**, implementando o protocolo *quórum consensus* para replicar a procura e compra de produtos por cada utilizador.

O sistema deverá suportar múltiplos clientes concorrentes. Cada cliente é identificado univocamente por um número inteiro, que se assume previamente atribuído e conhecido pelo próprio cliente.

Como modelo de interação e faltas, deve assumir-se que:

- Os gestores de réplica podem falhar silenciosamente, mas não arbitrariamente (i.e., não há falhas bizantinas);
- No máximo, existe uma minoria de gestores de réplica em falha em simultâneo;
- O conjunto de gestores de réplica é estático e os seus endereços são conhecidos *a priori*;
- O sistema é assíncrono e a comunicação pode omitir mensagens (apesar do projeto usar HTTP como transporte, deve assumir-se que outros protocolos de menor fiabilidade podem ser usados).

O sistema replicado deve contar com um conjunto estático de N gestores de réplica. Cada gestor de réplica deverá manter uma cópia local do estado do serviço *Supplier*.

A operação de escrita replicada corresponde à operação de definir o preço de um produto (*setProductPrice*) e a operação de leitura corresponde à operação de consultar o preço (*getProductPrice*).

É desejável que, tanto quanto possível, a solução assegure transparência de replicação e as opções tomadas privilegiem o desempenho da resposta ao cliente.

## 4 ENTREGA E AVALIAÇÃO

---

O trabalho de época especial é realizado individualmente.

A primeira parte vale 8 valores em 20, a segunda parte vale 5 valores e a terceira parte vale 7 valores.

A entrega do projeto será feita através do Fénix de acordo com as instruções a disponibilizar na página. A data limite é segunda-feira, dia 24 de julho de 2017, às 12:00.

A demonstração e discussão do projeto está prevista decorrer no mesmo dia, a partir das 13:00. O horário definitivo será anunciado no Fénix.

Para qualquer esclarecimento sobre o projeto deve contactar:

[leic-sod@disciplinas.tecnico.ulisboa.pt](mailto:leic-sod@disciplinas.tecnico.ulisboa.pt)

Os materiais de apoio ao projeto estão na seguinte página Web:

<http://disciplinas.tecnico.ulisboa.pt/leic-sod/2016-2017/labs/>

Votos de um bom trabalho!