DEI
DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
TÉCNICO LISBOA

2017

Komparator

PROJETO DE SISTEMAS DISTRIBUÍDOS (SD)

DATA DE REVISÃO: 2017-04-23

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

1 CENÁRIO

Os consumidores portugueses aderem cada vez mais às compras on-line[1]. Na sua experiência de compras deparam-se muitas vezes com lojas alternativas que oferecem os mesmos produtos. No entanto a informação apresentada não possui uma estrutura bem definida o que torna difícil a comparação de preços e de outras características.

O ***Komparator*** surge assim como um *serviço de mediação* de venda de produtos que faz uso de diversos fornecedores para satisfazer os pedidos dos seus clientes[2]. Assim, e num dado instante, o inventário global do serviço de mediação corresponde à união dos inventários de todos os fornecedores cujos serviços estão disponíveis. Um cliente tem a possibilidade de pesquisar o inventário global do serviço de mediação e de realizar pedidos de compra de múltiplos produtos, de diversos fornecedores.

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

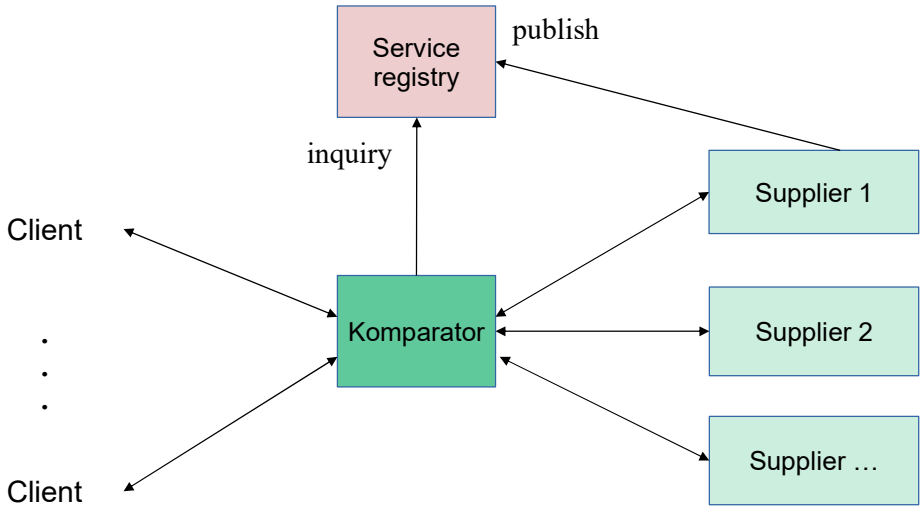

Figura 1 – Arquitetura do sistema.

Cada *fornecedor* tem um repositório de produtos, disponibilizando um serviço que permite efetuar pesquisas e compras de produtos. O *mediador* está ligado aos serviços dos fornecedores que conhece e faz procuras no conjunto dos inventários. Faz também a intermediação da compra.

Cada cliente seria uma aplicação que interagiria diretamente com o utilizador, aceitando comandos e apresentando resultados. No entanto, dado o âmbito da unidade curricular de Sistemas Distribuídos, os clientes são representados por baterias de testes.

O registo de serviços é utilizado para registar os diversos serviços (mediador e fornecedores). Os clientes consultam o registo para localizar os serviços. Por exemplo, o serviço de mediação deverá aceder a este registo quando quiser localizar fornecedores para contactar.

1 https://www.ecommerce-europe.eu/research-figure/portugal/

2 Inspirado em https://www.kuantokusta.pt/

2 PLATAFORMA DE DESENVOLVIMENTO

Todos os componentes do projeto serão implementados na plataforma Java, usando o JDK[3]. A construção de projetos deverá utilizar a ferramenta Maven.

A invocação remota de operações deve ser suportada por Web Services construídos com JAX-WS[4]. 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*[5] 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[6]. 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.

3 REQUISITOS

O projeto está organizado em quatro 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.

As operações estão descritas de forma detalhada no seguinte contrato WSDL[7]:
http://disciplinas.tecnico.ulisboa.pt/leic-sod/2016-2017/labs/proj/supplier.1_0.wsdl

3.1.1 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 Java Developer Kit – máquina virtual Java e respectivas ferramentas de desenvolvimento
4 Java API for XML Web Services – implementação de Web Services SOAP em Java
5 Integration Test – IT – de acordo com a convenção maven, estes testes executam-se na fase mvn verify
6 Universal Description, Discovery, and Integration; http://uddi.xml.org/
7 Web Services Description Language – documento que descreve as operações de um serviço, respetivos argumentos, resultados e erros

As validações de argumentos devem ser feitas pela ordem da sua definição, ou seja, deve ser validado o primeiro argumento em primeiro lugar, o segundo em segundo lugar e assim sucessivamente.

As operações são sensíveis à capitalização dos caracteres (são *case-sensitive*). Os identificadores de produto e de compra são alfa-numéricos e sem espaços.

3.1.2 Operações auxiliares

Estas operações destinam-se a auxiliar os testes e não necessitam de ser testadas exaustivamente.

- A operação *ping* responde com uma mensagem de diagnóstico não vazia. Sugere-se que a mensagem inclua a identificação do fornecedor.
- A operação *clear* apaga todo o estado do serviço.
- A operação *createProduct* recebe os dados de um produto, incluindo o identificador, a descrição e a quantidade, e cria o respetivo produto. O identificador e a descrição não podem ser vazios. A quantidade deve ser maior do que zero.
- A operação *listProducts* devolve uma lista com o estado de todos os produtos.
- A operação *listPurchases* devolve uma lista com todas as compras já efetuadas, ordenadas por ordem cronológica, da mais recente para a mais antiga.

3.2 MEDIADOR

O serviço mediador (mediator-ws) disponibiliza operações de agregação de vários fornecedores.

As operações do serviço mediador estão descritas no seguinte contrato WSDL, que descreve as operações, respetivos argumentos, resultados e erros:
http://disciplinas.tecnico.ulisboa.pt/leic-sod/2016-2017/labs/proj/mediator.1_0.wsdl

Nota: por simplificação, a funcionalidade pedida não inclui a realização do pagamento, apenas a validação do meio de pagamento.

3.2.1 Operações principais

A operação *getItems* recebe um identificador de produto e devolve o produto com o preço em diferentes fornecedores. Cada produto contém o seu identificador, o identificador do fornecedor, a descrição, e o preço. Os resultados da operação são apresentados ordenados por preço (surgindo o mais barato em primeiro lugar).

A operação *searchItems* recebe um texto de pesquisa e devolve os produtos cuja descrição contenha o texto recebido. O resultado desta operação é a composição dos resultados das pesquisas aos fornecedores. Os resultados da operação são apresentados ordenados por identificador de produto (ordem alfabética crescente) e depois por preço (mais barato em primeiro lugar).

A operação *addCart* recebe um nome de carrinho de compras, um identificador de produto e de fornecedor, e uma quantidade pretendida (número positivo). O produto deve ser colocado no carrinho de compras. Caso o carrinho de compras com o nome indicado não exista, é criado nesse momento. Caso o produto já exista no carrinho, deve ser adicionada a quantidade indicada. A quantidade pretendida de cada produto deve ser confirmada junto do fornecedor respetivo.

A operação *buyCart* recebe um nome de carrinho de compras e um número de cartão de crédito e efetua a compra de todos os produtos contidos nesse carrinho.

Antes de avançar com a compra, o serviço externo de validação de cartão de crédito (cc-ws) deve ser contactado para verificar que o cartão de crédito fornecido é válido.

Se o cartão de crédito for válido, o serviço mediador contacta os fornecedores necessários para efetuar a compra de cada produto, de forma sequencial. A operação de compra deve tentar alcançar o melhor resultado possível, ou seja, caso existam contrariedades na compra de produtos contidos no carrinho – por exemplo, fornecedor não contactável, quantidade insuficiente, etc. – a operação deve lidar com os problemas e prosseguir para o item seguinte.

Para cada item, apenas se aceita a compra na quantidade total pretendida, ou seja, ou se compra o item na quantidade pretendida ou não se compra nenhuma unidade do item.

Como resultado da operação de compra, é devolvido um identificador (gerado no momento), acompanhado de uma lista de produtos efetivamente comprados e outra lista com os produtos não comprados. O resultado apresenta também o preço total dos itens efetivamente comprados. Uma compra considera-se:

- COMPLETA se todos os itens pretendidos foram comprados;
- PARCIAL se pelo menos um item pretendido não foi comprado;
- VAZIA se nenhum item pretendido foi comprado.

As validações de argumentos devem ser feitas pela ordem da sua definição, ou seja, deve ser validado o primeiro argumento em primeiro lugar, o segundo em segundo lugar e assim sucessivamente.

As operações são sensíveis à capitalização dos caracteres (são *case-sensitive*). Os identificadores são alfa-numéricos e sem espaços.

3.2.2 Operações auxiliares

Estas operações destinam-se a auxiliar os testes e não necessitam de ser testadas exaustivamente.

- A operação *ping* responde com uma mensagem de diagnóstico não vazia. Sugere-se que o resultado desta operação inclua a concatenação do *ping* dos fornecedores conhecidos de modo a testar a conectividade de todo o sistema.
- A operação *clear* apaga todo o estado do serviço. Esta operação deve chamar a operação equivalente nos fornecedores, para limpar o estado de todo o sistema.
- A operação *listCarts* devolve uma lista com o estado de todos os carrinhos de compras. O estado de um carrinho corresponde a lista de produtos nele contidos.
- A operação *shopHistory* devolve uma lista com todas as compras já efetuadas, ordenadas por ordem cronológica, da mais recente para a mais antiga.

3.3 Segurança na comunicação Mediador-Fornecedores

Dada a natureza sensível dos dados trocados entre os clientes e o mediador, e entre o mediador e os fornecedores existem alguns requisitos de segurança, nomeadamente:

- O valor do cartão de crédito deve ser enviado do cliente para o mediador de forma a que apenas este último o consiga ler.
- As mensagens trocadas entre o mediador e os fornecedores devem estar autenticadas e não devem poder ser repudiadas *à posteriori* pelo seu emissor.
- As mensagens devem ser protegidas para garantir a sua frescura, i.e., devem prevenir ataques por repetição feitos por um agente malicioso que reenvia mensagens enviadas anteriormente por um agente legítimo.

Existe uma autoridade de certificação cuja chave pública é conhecida à partida por todos os outros intervenientes. Para alem disso, todos os intervenientes têm um certificado digital de chave pública emitido por essa autoridade de certificação. Apenas cada um conhece a sua própria chave privada.

Por simplificação, não será necessário proteger os restantes canais de comunicação, incluindo a comunicação entre o mediador e o serviço de validação de cartão de crédito, e com o UDDI.

3.4 Tolerância a Faltas do Mediador

Dada a centralidade neste sistema do mediador e a relevância da informação que guarda, deverá ser garantida a sua tolerância a falhas. Assume-se que a falta que pode ocorrer é a paragem do processo que implementa o mediador. Isso será conseguido replicando o mediador.

Existirá uma réplica do mediador que funcionará como servidor secundário alternativo. O mediador principal será responsável por garantir que o estado do servidor secundário é mantido atualizado de forma síncrona. O servidor secundário é responsável por detetar a falha do mediador principal e, nesse caso, por assumir o seu papel.

Não será necessário repor a replicação, ou seja, por simplificação, após a falta do servidor principal, passará a existir somente um servidor, que será o antigo secundário.

4 Entregas e Avaliação

As entregas do projeto serão feitas através do repositório Git de acordo com as instruções a disponibilizar na página Web.

Todos os alunos têm discussão final do projeto; nesta discussão qualquer aluno pode ter a sua nota obtida até então alterada. As notas das fases (P1 a P4) são indicativas e sujeitas a confirmação na discussão final na qual todo o *software* desenvolvido durante o semestre será tido em conta. As notas atribuídas são individuais.

4.1 Primeira Parte (P1)

Na primeira parte do projeto deverá ser implementado o serviço de fornecedor (conferir secção 3.1). Será fornecida uma estrutura base para o projeto.

A data limite de entrega é: *sexta-feira, 24 de março de 2017, 16:59*.

Esta parte do projeto vale 3 valores em 20, distribuídos da seguinte forma:

- Estrutura base (20%)
- Implementação das operações (60%)
- Vossos testes de integração (20%)

Esta parte do trabalho será avaliada através de uma bateria de testes de integração, disponibilizada após a entrega.

4.2 Segunda Parte (P2)

Na segunda parte deverá ser implementado o serviço de mediação (conferir secção 3.2), com pelo menos dois fornecedores associados.

A data limite de entrega é: *sexta-feira, 7 de abril de 2017, 23:59*.

Esta parte do projeto vale 5 valores em 20, distribuídos da seguinte forma:

- Estrutura base (10%)
- UDDI (10%)
- Múltiplas instâncias de fornecedores (10%)
- Implementação das operações (50%)
- Vossos testes de integração (20%)

Esta parte do trabalho também será avaliada através de uma bateria de testes de integração, disponibilizada após a entrega.

4.3 Terceira Parte (P3)

Na terceira parte deverá ser implementada a segurança (conferir secção 3.3).

A data limite de entrega é: *sexta-feira, 5 de maio de 2017, 23:59*.

Esta parte do projeto vale 6 valores em 20, distribuídos da seguinte forma:

- Estrutura base (5%)
    - Configuração (POMs e Handlers)
    - Código legível
    - Tratamento de exceções
    - Sincronização correta
- Testes P2 incorporados no projeto (20%)
    - getItems, searchItems, addToCart, buyCart
- Proteção canal client-mediator (15%)
    - Cifra (confidencialidade)
- Proteção canal mediator-supplier (45%)
    - Assinatura digital (autenticidade e integridade) e respetiva verificação
    - Frescura
    - Certificados (verificação e uso de CA externa)
- Documentação (15%)
    - Relatório
    - Guião de demonstração

Esta parte do trabalho será avaliada através de:

- Execução dos testes de integração do serviço mediador (testes da P2 já incorporados no código entregue pelo grupo) e;
- Análise ao registo de mensagens SOAP capturadas;
- Seguimento dos passos do guião de demonstração.

O *LoggingHandler* deve ser configurado para capturar mensagens, no cliente e no servidor, antes e depois de cada cadeia de processamento de mensagens SOAP, para se poder conferir as alterações efetuadas às mensagens SOAP:

- Cadeia de *handlers* no cliente: *LoggingHandler, YourHandler, LoggingHandler*
- Cadeia de *handlers* no servidor: *LoggingHandler, YourHandler, LoggingHandler*

O relatório de segurança deverá conter dois exemplos de mensagens capturadas – pedido e resposta – e apresentar uma legenda com breve explicação dos elementos XML incluídos para garantir os requisitos de segurança.

4.4 QUARTA PARTE (P4)

Na quarta parte deverá ser implementada a tolerância a faltas (conferir secção 3.4).

A data limite de entrega é: *sexta-feira, 19 de maio de 2017, 23:59*.

Esta parte do projeto vale 6 valores em 20, distribuídos da seguinte forma:

- Replicação (30%)
- Substituição de servidor (30%)
- *Front-end* cliente com semântica no-max-1-vez (20%)
- Documentação de tolerância a faltas (20%)
    - Relatório
    - Guião de demonstração

Esta parte do trabalho será avaliada através de uma demonstração ao vivo. Deverá ser preparado um guião para a demonstração.

5 NOTAS FINAIS

5.1 NOMES

O identificador do grupo tem o formato CXX, onde:

- C representa o campus (A para Alameda e T para Taguspark);
- XX representa o número do grupo de SD atribuído pelo Fénix.

Por exemplo, o grupo A22 corresponde ao grupo 22 sediado no campus Alameda; já o grupo T31 corresponde ao grupo 31 sediado no Taguspark.

Os nomes a usar para os serviços são: CXX_Mediator, CXX_Supplier1, CXX_Supplier2, etc. Ou seja, o grupo A22 deverá registar os seus serviços com o seguinte nome: A22_Mediator, A22_Supplier1, A22_Supplier2, etc. Quanto ao grupo T31, os nomes seriam: T31_Mediator, T31_Supplier1, etc.

5.2 SERVIÇOS EXTERNOS ALOJADOS

Alguns dos serviços necessários já se encontram implementados e devem ser utilizados remotamente:

- UDDI: http://uddi.sd.rnl.tecnico.ulisboa.pt:9090
- CC: http://ws.sd.rnl.tecnico.ulisboa.pt:8080/cc?WSDL
- CA: http://sec.sd.rnl.tecnico.ulisboa.pt:8081/ca?WSDL

5.3 RELATÓRIO

O relatório deve ser entregue juntamente com o projeto, na pasta `doc/` na raiz do projeto. O ficheiro deve chamar-se `CXX-relatorio-seguranca/tolfaltas.pdf`, consoante o caso.

O documento deve conter:

- 1 folha de rosto:
  - Identificador do grupo em formato CXX;
  - URL do repositório no GitHub;
  - Fotos, números e nomes dos membros do grupo (ordenado por número de aluno).
- 2 páginas de conteúdo:
  - Figura da solução de segurança/tolerância a faltas (deve ocupar 1/2 página);
  - Descrição da figura e breve explicação da solução;
  - No caso da segurança, esquema das mensagens SOAP (pedido e resposta) e respetiva legenda e explicação.
  - No caso da tolerância a faltas, deve ser detalhada a troca de mensagens do protocolo implementado.

5.4 GUIÃO DE DEMONSTRAÇÃO

Cada grupo deve preparar um guião de demonstração com casos de utilização que demonstram as melhores funcionalidades do trabalho desenvolvido. Os guiões não têm formato pré-definido e devem ser incluídos na pasta `doc/` na raiz do projeto em formato `PDF`. Os ficheiros devem chamar-se `CXX-guiao-seguranca/tolfaltas.pdf`, consoante o caso

O guião deve incluir instruções de instalação e configuração, que devem começar com a obtenção do código no repositório Git, devem dar indicações de compilação e de como proceder para fazer a demonstração preparada.

Seguem-se sugestões sobre como estruturar o guião, quer para a parte de segurança, quer para a parte de tolerância a faltas.

Segurança

Duração inferior a 5 minutos

- Caso S1: passos para demonstrar funcionamento normal
  - Obrigatório: configuração de *LoggingHandler* para imprimir mensagens SOAP *antes* e *depois* da proteção. Destaque dos elementos seguros nas mensagens capturadas.
- Caso S2: passos para demonstrar resistência a um ataque
  - Sugestão: simular a alteração não autorizada de mensagem, por exemplo, para mudar o valor de um preço devolvido por um fornecedor.
  - Sugestão: pré-definir um código de produto especial (e.g. produto com código "XPTO") que despoleta o comportamento de ataque.

Replicação
Duração inferior a 5 minutos

- Caso R1: passos para demonstrar funcionamento normal da replicação
    - Obrigatório: imprimir para a consola cada vez que é feita a propagação de alterações e a prova-de-vida
- Caso R2: passos para demonstrar tolerância a falta
    - Sugestão: usar operação *ping* com argumento especial (e.g. "kill") que causa paragem súbita do servidor, usando o System.exit()
    - Alternativa: causar a paragem súbita do servidor primário através de sigkill (CTRL-C)
    - Alternativa: desligar a interface de rede (no caso de uma demonstração com várias máquinas)

5.5 NOVIDADES

Para se manter a par de novidades sobre o projeto, consulte a seguinte página Web regularmente: http://disciplinas.tecnico.ulisboa.pt/leic-sod/2016-2017/labs/

Bom trabalho!