DEI
DEPARTAMENTO
DE ENGENHARIA INFORMÁTICA
TÉCNICO LISBOA

LEIC-A, LEIC-T, LETI, MEIC-T, MEIC-A

Sistemas Distribuídos

2º Semestre – 2015/2016

Enunciado do Projeto: Upa Transportes

1. Introdução

O objetivo do projeto de Sistemas Distribuídos é desenvolver um sistema baseado em Web Services, implementados na plataforma Java, para agregação de serviços de transporte de mercadorias à semelhança do que, por exemplo, a Uber efetua para os transportes de passageiros. O nome do serviço a construir é `Upa` e a Figura 1 mostra uma visão global dos componentes da solução.

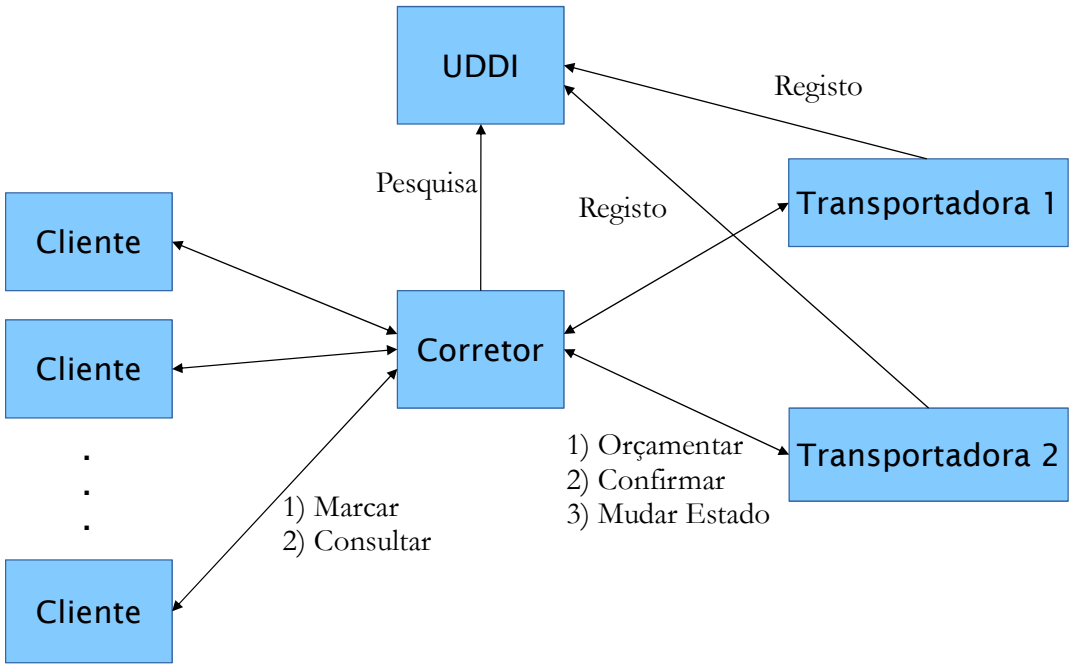

Figura 1: Arquitetura do projeto.

Neste sistema existem: os clientes, o corretor (serviço Upa que agrega ofertas) e as transportadoras (empresas de transportes). Os clientes interagem unicamente com o corretor a quem fazem pedidos de transporte. Os clientes podem depois consultar o estado dos seus pedidos.

O corretor está ligado aos serviços das transportadoras que conhece e procura entre as empresas disponíveis aquela que pode satisfazer o pedido do cliente pelo preço mais baixo. Uma vez selecionada a melhor oferta para um pedido, o corretor contacta a empresa que fez a oferta e atribui-lhe essa viagem, guardando um registo dessa marcação.

As transportadoras recebem pedidos de orçamento e de marcação de viagens. Aos pedidos de orçamento para uma dada viagem respondem com uma proposta de preço. Aos pedidos de marcação de viagem respondem com uma confirmação.

A Figura 2 mostra um resumo da evolução do estado de uma viagem, no corretor e na transportadora.

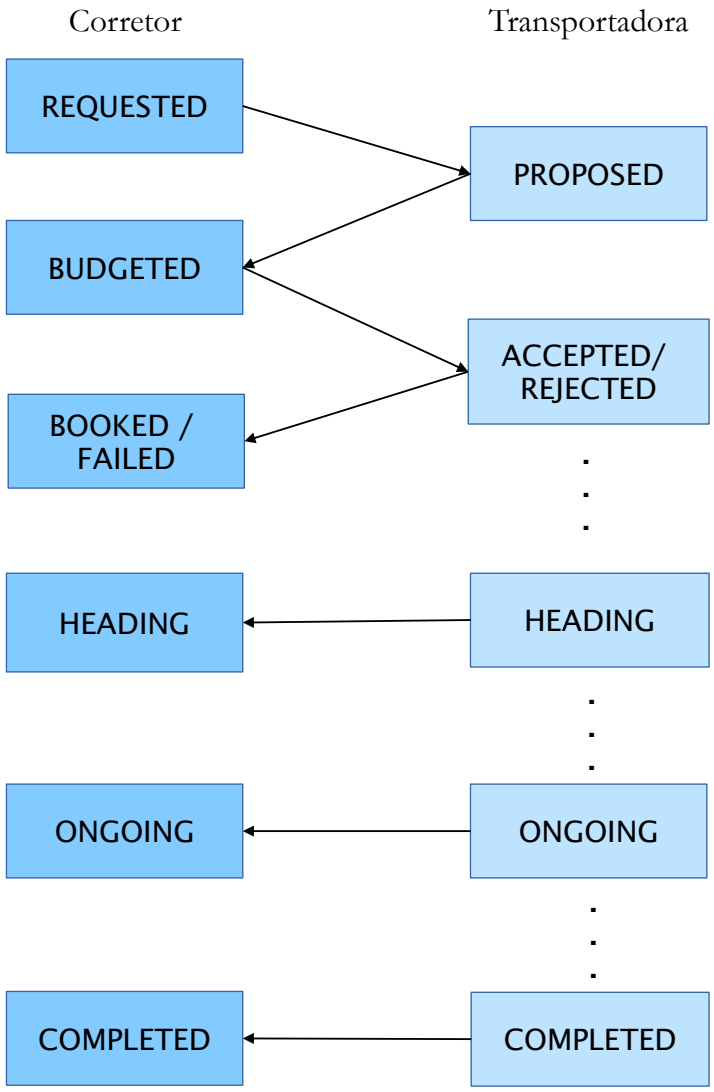

Figura 2: Estados do transporte no corretor e na transportadora.

2. Tecnologia

Todos os componentes do projeto serão implementados na linguagem de programação Java. A invocação remota de serviços deve ser suportada por Web Services usando JAX-WS.

Os serviços de transporte que competem entre si devem ser descobertos e localizados dinamicamente, por intermédio de um servidor jUDDI. Cabe a cada Web Service das transportadoras, quando lançado, registar-se a si próprio no servidor jUDDI bem conhecido, indicando o seu URL. É no servidor jUDDI que o corretor irá consultar a lista de todas as empresas de transportes para estabelecer contacto com os seus Web Services. Os nomes a usar para os serviços são: `UpaBroker`, `UpaTransporter1`, `UpaTransporter2`, etc.

2.1. Testes

Espera-se que cada projeto inclua testes unitários e de integração, que permitam cobrir todos os requisitos funcionais do enunciado. Deverão também ser testadas situações de falhas de comunicação, falha silenciosa do serviço ou falha do servidor de UDDI.

Para os *testes unitários* de cada componente, que verificam o comportamento dos objetos através de invocações *locais*, devem usar JUnit com JMockit para simular todos os servidores remotos (UDDI e outros Web Services).

Para os *testes de integração* (IT), que verificam o cumprimento do contrato de um Web Service através de invocações *remotas*, devem usar JUnit para invocar todos os servidores remotos necessários (que se assume terem sido previamente lançados de forma correta).

Para os testes de integração deverão ativar duas empresas transportadoras diferentes: `UpaTransporter1` e `UpaTransporter2`.

3. Clientes

As aplicações cliente representam as empresas ou particulares que querem marcar viagens. Para simplificar o trabalho e de acordo com os objetivos primordiais desta UC, não existe uma interface utilizador mas apenas testes em linha de comando.

Os clientes podem pedir a marcação de uma viagem e depois ir pedindo o estado da viagem ao corretor. Para além da possibilidade de marcar viagens, os clientes podem ainda listar as viagens registadas no corretor a fim de se poder testar se a lista de viagens se mantém correta.

4. Corretor

O corretor agiliza a interação entre clientes e empresas de transporte agregando as ofertas das transportadoras, encaminhando os pedidos dos clientes para as transportadoras, negociando o melhor preço para os clientes e mantendo um registo das viagens marcadas.

As operações do serviço do corretor são descritas no seguinte contrato WSDL, que descreve as operações, respetivos argumentos, resultados e erros; e que deverá ser respeitado:
`http://disciplinas.tecnico.ulisboa.pt/leic-sod/2015-2016/labs/proj/broker.1_0.wsdl`

4.1. Operações principais

A operação *requestTransport* tenta marcar um transporte de uma origem para um destino. O cliente define um preço máximo para o serviço. São lançados erros caso: a origem ou o destino seja desconhecido; o preço seja menor que 0; não exista um transporte disponível da origem para o destino; não exista um transporte disponível com o preço pretendido.

As origens e destinos reconhecidos pelo corretor são todos os seguintes:
Região Norte: Porto, Braga, Viana do Castelo, Vila Real, Bragança;
Região Centro: Lisboa, Leiria, Santarém, Castelo Branco, Coimbra, Aveiro, Viseu, Guarda;
Região Sul: Setúbal, Évora, Portalegre, Beja, Faro.

A operação *viewTransport* devolve uma vista do estado atual do transporte. Para saber o estado mais recente do transporte poderá ser necessário contactar a transportadora respetiva.

4.2. Operações auxiliares

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

A operação *ping* responde com uma mensagem de diagnóstico que não pode ser vazia. Sugere-se que nesta operação se tente estabelecer contacto com as transportadoras.

A operação *listTransports* devolve uma lista com o estado de todos os transportes.

A operação *clearTransports* apaga todos os transportes e pede às transportadoras para também apagarem todos os seus dados. Esta operação permite "limpar" todo o estado do sistema.

4.3. Progresso do Estado das Viagens no Corretor

As viagens registadas no corretor vão mudando de estado ao longo do tempo como apresentado na Figura 2. Os estados possíveis para uma viagem são:

- `REQUESTED` (Pedida): O cliente pediu a marcação de uma viagem.
- `BUDGETED` (Orçamentada): Existe uma oferta de uma empresa de transportes para a viagem, mas ainda não foi marcada.
- `BOOKED` (Marcada): A viagem foi marcada com sucesso junto da empresa de transportes selecionada.
- `FAILED` (Falhada): A viagem não foi marcada com sucesso.
- `HEADING` (A Caminho): O condutor e veículo que realizará o transporte estão a caminho do cliente.
- `ONGOING` (Em Curso): A viagem está a decorrer.
- `COMPLETED` (Concluída): A viagem terminou.

As viagens são inseridas no registo do corretor no estado `REQUESTED`. O corretor pede orçamentos a todas as empresas. Caso não receba pelo menos um valor abaixo de `PrecoMax`, a viagem passa ao estado `FAILED`. Caso receba pelo menos um valor abaixo de `PrecoMax`, o corretor aprova a oferta mais barata e rejeita as outras; a viagem passa então para o estado `BOOKED`. Caso exista um empate nas ofertas, o corretor pode escolher qualquer uma das propostas recebidas com menor valor.

5. Empresas de Transporte

O serviço de transporte é implementado por um servidor que fornece um conjunto de serviços próprios de uma empresa de transportes. As empresas de transporte mantêm um registo de todas as viagens que orçamentam e aprovam.

As operações do serviço do transportador são também descritas no seguinte contrato WSDL, que descreve as operações, respetivos argumentos, resultados e erros:
`http://disciplinas.tecnico.ulisboa.pt/leic-sod/2015-2016/labs/proj/transporter.1_0.wsdl`

5.1. Operações principais

A operação *requestJob* devolve uma proposta de preço para o transporte pedido. São lançados erros caso: a origem ou o destino seja desconhecido; o preço seja menor que 0. As transportadoras com número ímpar (ex. `UpaTransporter1`) operam nas regiões Centro e Sul; as transportadoras com número par (ex. `UpaTransporter2`) operam no Centro e Norte.

A operação *requestJob* recebe um preço de referência, mas é livre de retornar qualquer valor. Caso não esteja interessada na proposta ou não opere na origem ou destino, deve devolver `null`. Para poder recriar diferentes situações, cada transportadora deve responder aos pedidos que recebe dentro das regiões em que opera, de acordo com as seguintes regras:

Caso o preço de referência recebido seja maior do que 100, a transportadora não deve fazer oferta (devolve `null`). Caso o preço de referência seja menor ou igual a 10, a transportadora deve sempre fazer uma oferta abaixo do preço.

Para os restantes valores (preço de referência maior do que 10 e menor ou igual a 100):
caso o preço seja um número *ímpar*, uma transportadora com número também *ímpar* (ex. `UpaTransporter1`) faz uma oferta abaixo do preço e acima do preço em caso contrário;
caso o preço seja um número *par*, uma Transportadora com número também *par* (ex. `UpaTransporter2`) faz uma oferta abaixo do preço e acima do preço em caso contrário.

A operação *decideJob* deve ser chamada para aceitar/rejeitar uma proposta, sendo feita a transição de estado respetiva.

A operação *jobStatus* devolve uma vista com a informação disponível sobre a viagem. Caso o identificador não seja válido, deve devolver `null`.

5.2. Operações auxiliares

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

A operação *ping* responde com uma mensagem de diagnóstico, que não deve ser vazia.

A operação *listJobs* devolve uma lista de todas as viagens.

A operação *clearJobs* deve apagar todas as viagens.

5.3. Simulação dos transportes

A evolução do estado da viagem é feita no transportador, através de temporizadores.

Uma viagem no estado `ACCEPTED` passará ao estado `HEADING` após ter decorrido um intervalo de tempo aleatório com uma duração entre 1 e 5 segundos (por exemplo, pode ser sorteado um intervalo de 2207 milissegundos). Seguidamente, a viagem passará ao estado `ONGOING` após uma nova duração aleatória (outro valor sorteado entre 1 e 5 segundos). Por fim, a viagem passará ao estado `COMPLETED` após mais uma duração aleatória (ainda outro valor sorteado entre 1 e 5 segundos).

6. Segunda Parte do Projeto

Na segunda parte do projeto serão implementados dois requisitos não-funcionais que aumentarão o realismo do projeto, nomeadamente a segurança na negociação de viagens e a tolerância a falhas do corretor.

6.1. Segurança na Comunicação Corretor-Transportadoras

Dada a natureza sensível dos dados trocados entre o corretor e as empresas de transportes existem alguns requisitos não-funcionais de segurança. As mensagens trocadas entre o corretor e os serviços de transporte devem estar autenticadas e não devem poder ser repudiadas *a posteriori* pelo seu emissor.

As mensagens devem ser protegidas para garantir a sua frescura i.e. que não são repetições feitas por um agente malicioso de mensagens velhas enviadas anteriormente por um agente legítimo.

Assume-se que existe uma autoridade de certificação, cuja chave pública é conhecida à partida de todos os intervenientes. Para além 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.

6.2. Tolerância a Faltas do Corretor

Dada a centralidade neste sistema do corretor 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 corretor. Isso será conseguido replicando o corretor.

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

Não é necessário repor a replicação, ou seja, por simplificação, após a falha do servidor principal, passará a existir somente um servidor: que será o antigo secundário transformado em novo corretor principal.

7. Entrega e Avaliação

A entrega do projeto será feita através do repositório `Git` de acordo com as instruções a disponibilizar na página Web. Os serviços entregues deverão ter dados de teste pré-carregados que permitam demonstrar todas as funcionalidades.

7.1. Primeira parte

A primeira parte do projeto deve ser realizada até à data de entrega intermédia no dia 15 de abril de 2016 às 23:59. A entrega corresponde a 8 valores em 20 da nota do projeto.

Os itens que serão avaliados na primeira entrega são:

- Transportadoras (total 40%)
  - Estrutura base (10%)
  - UDDI (5%)
  - Implementação das operações (10%)
  - Vossos testes (10%)
  - Múltiplas instâncias (5%)
- Corretor (total 60%)
  - Estrutura base (15%)
  - UDDI (5%)
  - Implementação das operações (20%)
  - Vossos testes (20%)

7.2. Segunda parte

A segunda parte do projeto corresponde a 12 valores em 20. O prazo de entrega final do projeto é no dia 13 de maio de 2016 às 23:59.

Os itens que serão avaliados na segunda entrega são:

- Servidores funcionais, com UDDI, e testes de integração (20%)
- Segurança (30%)
- Replicação com primário-secundário (40%)
- Relatório e Demonstração final (10%)

Consulte a página Web regularmente para se manter a par de novidades sobre o projeto:

`http://disciplinas.tecnico.ulisboa.pt/leic-sod/2015-2016/labs/`

Bom trabalho!