Instituto Superior de Engenharia de Lisboa

Sistemas Distribuídos

Relatório da terceira série

Autoria:
32632 Pedro Pedroso
33724 David Raposo
33404 Ricardo Mata
5 de Julho de 2014

Conteúdo

1 Introdução

Neste terceiro trabalho da disciplina de Sistemas Distribuídos, foi requerido que se implementasse as diferentes componentes de um sistema de pesquisa de automóveis e respetiva reserva.

Este sistema é composto pelos seguintes intervenientes:

- Cliente: Aplicação gráfica onde serão feitas as pesquisas e respetivos pedidos de reserva.
- Stands: Aplicações que contêm um conjunto de carros, onde incidem as pesquisas e reservas dos clientes. Mantém dados persistentes usando ficheiros *XML* para obter/guardar a informação necessária.
- Broker: Aplicação que o cliente comunicará para fazer a sua pesquisa e onde os *stands* se podem registar para mais tarde ser propagada as pesquisas dos clientes.

---

2 Arquitetura

(Adicionar imagem da arquitetura)

(Explicação do flow da informação. Assim como são passados os endpoints do cliente para o stand e do stand para o cliente)

As diversas aplicações foram desenvolvidas em *c-#*. Na componente gráfica do cliente foi usado *Windows Forms*, enquanto para a componente *web* das diversas aplicações foi usado *WCF*.

Para o desenvolvimento dos serviços, existem um conjunto de restrições, sendo estas o uso de *Binding wsHttpBinding* durante a comunicação entre

3 Componentes

3.1 *Broker*

Componente intermédio para fazer *broadcast* ás pesquisas dos clientes para os stands e para registar os mesmos.

3.1.1 Serviços disponibilizados

- *IBrokerClientService:* Serviço que permite aos clientes submeterem as suas pesquisas ao *Broker*.
  - Modo de instanciação: Singleton
    Visto não haver alterações de objetos neste serviço e admitindo que haverá vários clientes a fazer várias pesquisas então não seria razoável criar um objeto por chamada. Um só objeto consegue lidar com os pedidos de forma eficaz.
  - Modo de controlo de concorrência: Multiple
    Ao receber vários pedidos será razoável que o processamento desses pedidos seja *multi-thread*. Visto não haver alterações nenhumas em váriaveis no *Singleton* então não é necessário sincronismo adicional por parte do *Broker*.
  - Binding utilizado: basicHttpBinding
    Visto a informação enviada para o *Broker* não ter dados importantes, é utilizado *basicHttpBinding* para simplificar a mensagem enviada. Visto também que o cliente não obtém *feedback* nesta comunicação, os métodos estão declarados como *IsOneWay*. Esta opção faz com que o *emph* seja imediatamente libertado após a receção da mensagem por parte do *Broker* e liberta este *binding* para ser utilizado pelos stands para submissão de propostas.
- *IBrokerStandService:* Serviço que permite que um *stand* se registe no *Broker* para receber as pesquisas dos clientes.
  - Modo de instanciação: Singleton (What to say here....)
  - Modo de controlo de concorrência: Multiple Com a possibilidade de vários stands se registarem ao mesmo tempo, utilizamos um modo de concorrência *Multiple*. Isto cria problemas em relação ao sincronismo do objeto que está a ser usado para guardar os *proxies* que é resolvido (david preciso que expliques aqui o método que usaste para resolver esta concorrencia)
  - Binding utilizado: wsHttpBinding Binding utilizado como requisito não funcional.

3.2 *Client*

Componente que faz a submissão das propostas e reserva dos carros. O cliente tem uma interface gráfica para melhor ajudar a visualização das propostas e submissão das várias pesquisas.

3.2.1 Serviços disponibilizados

- *IClient:* Serviço que permite aos stands submeteram as propostas aos clientes.
  - Modo de instanciação: PerCall
    Um cliente só receberá propostas depois de efetuar uma pesquisa. Isto faz com que usar um objecto Singleton não seja razoável no sentido que não há necessidade de manter um objeto de serviço alocado quando não se espera mais propostas. Por esta razão foi escolhido o método de instanciação *PerCall* que cria um objeto de serviço que tem a duração da submissão da proposta.
  - Modo de controlo de concorrência: Multiple
    (como é que o updateTextBox no submitProposal está a funcionar threadsafe se isto está com multiple?)
  - Binding utilizado: basicHttpBinding
    Binding utilizado como requisito não funcional.

3.3 *StandAuto*

Componente que faz guarda a base de dados de carros e submete as propostas aos clientes com base nas pesquisas feitas.

3.3.1 Serviços disponibilizados

- *IStandBrokerContract:* Serviço que permite ao *Broker* propagar a pesquisa aos *stands*.
  - Modo de instanciação: Singleton
    Admitindo que este serviço irá estar frequentemente a receber pedidos vindos do *Broker* é razoável a utilização de um objeto de serviço *Singleton*.
  - Modo de controlo de concorrência: Multiple
    Vários pedidos implicam um ambiente *multi-thread* portanto é usado o controlo de concorrência *multiple*. Não é alterado nenhum objeto no *stand* portanto não é necessário controlo de concorrência adicional.
  - Binding utilizado: wsHttpBinding
    Binding utilizado como requisito não funcional.

3.3.2 Base de dados

(Explicação do processo de storage da base de dados de modo persistente)

4 Concorrência

Nos diversos serviços criados, problemas de concorrência apenas poderiam surgir no *Broker* (devido ao *broadcast* das pesquisas poder concorrer com adições/remoções de stands) e nos *Stands* (já que pode ser desejável evitar a propagação de propostas para carros já reservados). Em ambos os casos optou-se por permitir que

os serviços em si sejam concorrentes, o que permitiu que ambos serviços fossem *Singletons*.

As implementações *thread-safe* foram diferentes em cada situação, para resolver a situação da adição de *Stands* enquanto pesquisas estão a decorrer, optou-se por usar mecanismos de *lock*, já que não foi encontrada nenhuma coleção concorrente que suportasse remoção adequada. Enquanto no caso das reservas, optou-se por permitir que pesquisas obtivessem resultados que talvez não tenham a sua reserva registada, assegurando apenas que não é possível múltiplas reservas no mesmo carro.

5 Conclusões