3º Trabalho Prático de Avaliação

Objectivo: Desenvolvimento de sistemas distribuídos usando arquitecturas baseadas em serviços na plataforma .NET com ***Windows Communication Foundation (WCF)***

Nota: O trabalho deve ser realizado até 19 de Junho de 2017, incluindo um relatório que descreva o trabalho com as opções tomadas ao longo da sua realização. (Enviar, os projectos em Zip file e relatório, para lass@isel.ipl.pt). Note que, como foi referido na apresentação da disciplina, a qualidade do relatório terá peso na avaliação do trabalho realizado. O relatório deverá permitir ao leitor entender a arquitectura do sistema, bem como os aspectos relevantes da implementação dessa arquitectura realçando os pontos fortes e fracos da solução do problema. O relatório deve descrever em detalhe como foi implementado o mecanismo de *multicast* com garantia de ordenação causal de mensagens. Evite descrever código a menos que se justifique nalguma situação especial, por exemplo os contratos dos serviços, bem como os ficheiros de configuração dos serviços.

Considere um cenário de um sistema distribuído ilustrado no diagrama da figura 1 e descrito de seguida:

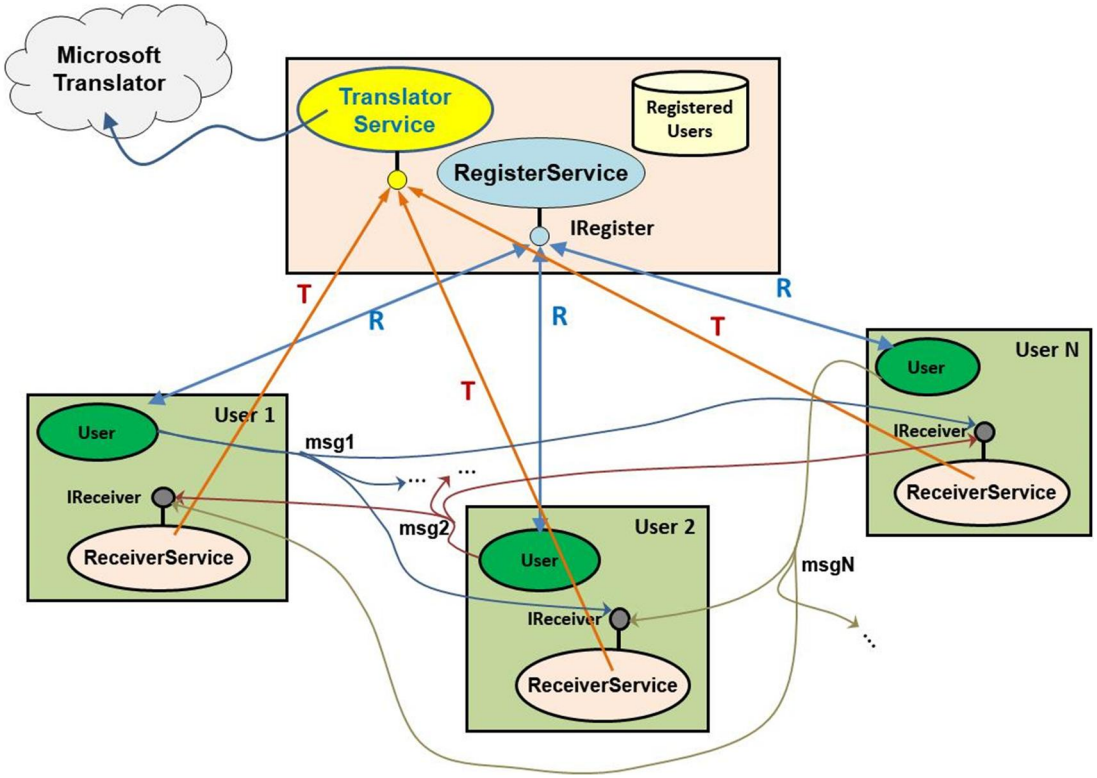

Figura 1 – Diagrama do sistema

Existe um Servidor que aloja dois serviços com *EndPoints* bem conhecidos (*TranslatorService*, um serviço de tradução de texto baseado no serviço externo *Microsoft Translator* e *RegisterService* e um serviço de registo de utilizadores). O servidor serve de auxiliar no processo de vários utilizadores se possam conhecer por forma a enviarem entre si mensagens de texto em múltiplas línguas.

Existe uma aplicação *User* que para além de alojar um serviço de receção de mensagens (*ReceiverService*) permite aos vários utilizadores enviar e receber mensagens *multicast* com garantia de ordenação causal.

Interações:

R: Os utilizadores registam-se ou terminam o seu interesse em receber mensagens (*Unregister*), bem como obtêm a lista de utilizadores registados num determinado momento e para os quais podem posteriormente enviar mensagens multicast. No registo é também indicado o endereço do *Endpoint* do serviço *ReceiverService*. Por questões de simplicidade o estado dos utilizadores registados é armazenado em memória;

T: O serviço de receção de mensagens traduz o texto da mensagem para a língua nativa do utilizador recorrendo ao serviço Translator que deverá ser um *wrapper* do serviço Microsoft Translator disponibilizado na página http://www.microsofttranslator.com/ e passível de ser invocado como Web Service no URL http://api.microsofttranslator.com/V2/soap.svc. Caso o serviço da Microsoft não esteja disponível, o *Translator Service* retorna a mensagem sem a traduzir. O serviço controla o acesso através de um *AppID* que pode ser obtido em http://www.bing.com/developers ou caso não obtenham um *AppID* pessoal, podem usar "F4E6E0444F32B660BED9908E9744594B53D2E864";

msg1, msg2, msgN: As mensagens multicast devem ser enviadas e recebidas com garantia de ordenação causal. Por exemplo, na figura 1, se o *User2* envia a mensagem *msg2* após ter lido a mensagem *msg1*, então o *UserN* deve receber primeiro *msg1* e só depois *msg2*, dado que *msg2* tem dependência causal de *msg1*.

Pressupostos e requisitos não funcionais:

- ✓ Assuma que o servidor nunca falha e que um *User* faz sempre *Unregister*, isto é, nunca falha enquanto estiver registado;
- ✓ Para implementar a ordenação causal de mensagens use o mecanismo de relógios lógicos vetoriais em que o vetor de relógios é passado, nas mensagens SOAP do serviço *ReceiverService*, como *Soap Header*. Isto é o serviço de receção *ReceiverService* deve usar um *Message Contract*. Para implementar o mecanismo *multicast* assuma que antes de cada mensagem a aplicação *User* valida junto do servidor quem está ou não presente;
- ✓ Deve utilizar ficheiros de configuração, simplificando assim a construção de um protótipo de demonstração com pelo menos 3 utilizadores;
- ✓ Tenha em atenção o tratamento e propagação de exceções para assim o sistema ser mais fiável e permitir tratar as falhas (utilize *Fault Contracts*). Note que embora se assuma que o servidor e as *Apps User* nunca falham poderão existir outro tipo de falhas.

Sugestões:

1. Qualquer questão ou dúvida sobre requisitos, deve ser discutida com o professor;
2. Antes de começar a escrever código desenhe a arquitectura do sistema, os contratos dos serviços bem como os diagramas de interacção mais importantes;
3. No relatório descreva e justifique as opções tomadas, por exemplo, o modo de instanciação e controlo de concorrência dos serviços, bem como a forma como na aplicação *User* é feita a interação entre a componente de "user interface" com o serviço de receção de mensagens;
4. Quando tiver questões sobre os requisitos, verifique no site *Moodle* se existem "*Frequently Asked Questions*" com esclarecimentos sobre o trabalho.

Luís Assunção