1º Trabalho Prático de Avaliação

Nuno Veloso 42181
Steven Brito 42798
Daniela Gomes 42799

3 de Maio de 2018

Conteúdo

Capítulo 1

Introdução

Com o intuito de realizar o 1º trabalho prático da unidade curricular Sistemas Distribuídos pretende-se implementar um sistema distribuído usando objetos distribuídos da plataforma .NET. Este sistema terá de ser capaz de trocar mensagens textuais e instantâneas entre utilizadores dentro da mesma região ou entre regiões. Será possível também formar grupos e trocar mensagens entre estes.

Ao longo deste trabalho iremos aplicar os conceitos aprendidos nas aulas. Iremos descrever e discutir as vantagens, os problemas e os desafios que se colocam no desenvolvimento deste sistema distribuído.

Capítulo 2

Descrição do Problema

O problema consiste em desenvolver um sistema distribuído capaz de trocar mensagens instantâneas, ou seja, sem haver o armazenamento das mesmas, entre utilizadores dentro da mesma região ou entre regiões. Há também a possibilidade de criar e remover grupos contendo utilizadores da mesma ou diferentes regiões. A troca de mensagens deve ser possível entre grupos. É também necessário garantir que os vários acessos simultâneos aos servidores mantenham a consistência dos dados. O sistema desenvolvido deve ter em conta tolerância a falhas, assim como a sua escalabilidade.

Capítulo 3

Requisitos

Neste capitulo é descrito os requisitos funcionais do sistema, assim como os não funcionais.

3.1 Funcionais

Os requisitos funcionais deste sistema são os seguintes apresentados:

- Evitar sobrecarga no servidor central;
- O cliente comunica apenas com o servidor da sua região;
- O cliente tem um identificador único;
- O cliente conhece todos os servidores regionais;
- O cliente regista-se num servidor regional;
- O cliente pode enviar mensagens para um único cliente ou para um grupo onde este pertença;
- O cliente pode mudar de região;
- O cliente pode criar grupos;
- O grupo pode ter clientes pertencentes a várias regiões;
- Os servidores regionais e centrais devem conhecer a estrutura dos grupos;
- Os clientes que não estejam conectados não recebem as mensagens.

3.2 Não Funcionais

Os requisitos não funcionais deste sistema são os seguintes apresentados:

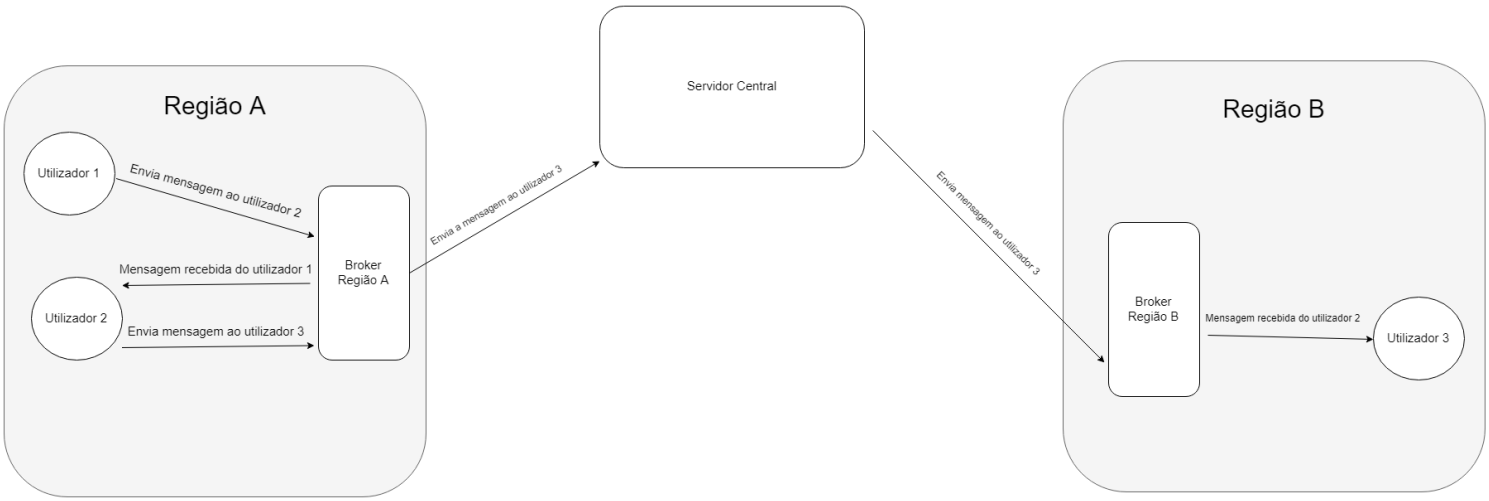
- A aplicação cliente realizada em WinForm;
- Ter um sistema escalável;
- Transparência à concorrência no acesso dos servidores;
- Suportar vários canais de comunicação (e.g.: HTTP, TCP);
- Tratar as falhas de forma a que o sistema continue funcional, mesmo com a existência de erros em um dos componentes.

Capítulo 4

Arquitetura

4.1 Interação entre as partes

Um utilizador apenas interage com o servidor regional onde este está registado. O servidor regional comunica-se com o utilizador e com o servidor central. O servidor central apenas interage com os servidores regionais.



O utilizador é cliente do servidor regional. O servidor regional é servidor do utilizador e cliente do servidor central. O servidor central é servidor do servidor regional.

4.2 Funcionamento

Para poder utilizar o sistema, o utilizador deve escolher em qual região pretende-se conectar. Estando este conectado, é possível usufruir das seguintes funcionalidades:

- Enviar mensagens para o utilizador;
- Criar grupos;
- Enviar mensagens para um grupo;
- Trocar de região;
- Sair de uma região.

4.2.1 Funcionamento entre utilizadores

Para um utilizador enviar uma mensagem para outro utilizador da mesma região o processo ocorre apenas no servidor dessa região. Caso o utilizador deseje enviar uma mensagem para outro utilizador fora da sua região, o servidor da região irá comunicar-se com o servidor central dizendo para este tratar de enviar a mensagem para um determinado utilizador que o servidor regional não tem conhecimento. O servidor central irá enviar a mensagem a todos os servidores regionais que conhece e estes irão verificar se o utilizador encontra-se registado na respetiva região. Em caso afirmativo tratarão de enviar a mensagem para o utilizador pretendente. Em caso negativo, o servidor regional não comunica com nenhum utilizador.

4.2.2 Funcionamento em grupo

É possível ao utilizador criar grupos dentro de uma região e adicionar outros utilizadores quer estejam na mesma ou em diferentes regiões. A existência do grupo é replicada por todos os outros servidores regionais. Ao adicionar um utilizador de outra região, o servidor regional envia a informação ao servidor central, que por sua vez envia aos outros servidores regionais até que a região onde se encontra o utilizador receba a mensagem e o adiciona ao grupo. Aos utilizadores que pertençam a um grupo, estes têm permissão para adicionar outros utilizadores. Os utilizadores que pertencem a um grupo nunca podem sair desse grupo, só sairão quando o criador apagar o grupo.

4.3 Escalabilidade

O sistema é escalável horizontalmente. Desta forma, se um dos servidores regionais ficar sobrecarregado, é possível adicionar mais servidores regionais.

Capítulo 5

Implementação

A solução encontra-se dividida em vários projetos de bibliotecas, aplicações de consola e uma aplicação WinForm.

5.1 Central Manager

Para ser implementado o servidor central, foi necessário definir a interface ICentralManager, apresentada na figura 5.1.

```
public interface ICentralManager
{
    void RegisterGroup(Group group, IBroker callerBroker);

    void AddUserToGroup(string groupName, int destNumber, IBroker callerBroker);

    void SendMessageToBrokers(int receiver, Message message, IBroker callerBroker);

    void SendMessageToGroup(string groupName, Message message, IBroker callerBroker);

    void UnregisterGroup(string groupName, IBroker callerBroker);
}
```

Figura 5.1: Interface ICentralManager

O *central manager* conhece todos os servidores regionais existentes, pois é injetado no construtor uma lista com todos os *proxies* dos servidores regionais desse sistema. Desta forma, o *central manager* é *stateless* visto que não armazena mais nenhuma informação.

Para a implementação do *central manager*, foi usado o padrão *singleton*. O padrão *singleton* adequa-se melhor às necessidades, pois assume-se que o servidor central estará sempre a receber e a enviar pedidos. Desta forma o padrão *single call* teria muito maior *overhead* em relação ao *singleton* pois teria de estar sempre a criar novas instâncias para responder a cada pedido, sendo que não se sabe o número de pedidos de antemão.

De modo a que não é possível determinar o uso deste sistema distribuído, não existe forma de saber se é usado com bastante regularidade e com espaçamento no intervalo de tempo entre as mensagens; No pior dos casos tem o objeto em memória com tempo infinito e os pedidos nunca passam pelo *manager*, e no melhor dos casos os pedidos são atendidos sempre pela

mesma instância, sem esta estar a ocupar recursos em memória desnecessariamente.

O *lease time* escolhido para a instância do *central manager* foi zero, o que significa que o seu tempo de vida é infinito. Foi tomada esta decisão porque a implementação de *Central Manager* não tem nenhum construtor sem parâmetros, pelo que o CLR não conseguiria criar uma nova instância, caso esta tenha sido apagada. O inconveniente desta solução é ter recursos ocupados sem haver necessidade de os ter, uma vez que esta instância pode não estar a servir pedidos.

5.2 Broker

Para implementar o servidor regional, definiu-se duas interfaces. A interface para interação com o utilizador é IBrokerService, apresentada na figura 5.2.

```
public interface IBrokerService
{
    void Register(IUser user);

    void RegisterGroup(int userNumber, string groupName);

    void AddUserToGroup(string groupName, int destNumber, int srcNumber);

    void SendMessageToUser(int destUserNumber, string message, int srcUserNumber);

    void SendMessageToGroup(string groupName, string message, int srcUserNumber);

    void Unregister(int userNumber);

    void UnregisterGroup(string groupName, int userNumber);
}
```

Figura 5.2: Interface IBrokerService

A interface para a interação com o *central manager* é IBrokerClient, apresentada na figura 5.3.

O *broker* mantém informação dos utilizadores registados nessa região e dos grupos existentes em todas as regiões. Desta forma, o *broker* é *stateful*.

Para a implementação do *broker* foi usado o padrão *singleton*. Foi usado este padrão, pois existe a necessidade de manter estado. Se fosse usado o padrão *single call*, tal não era permitido porque seria criado uma instância por cada pedido, impossibilitando a agregação de informação.

O *lease time* escolhido foi zero, o que significa que este terá um tempo de vida infinito. Isto porque devido à necessidade de manter estado, se fosse chamado o desconstrutor da instância, perder-se-ia toda a informação sobre os utilizadores e grupos.

```
public interface IBrokerClient
{
    void RegisterGroup(Group group);

    void AddUserToGroup(string groupName, int destNumber);

    void SendMessageToUser(int receiver, Message message);

    void SendMessageToGroup(string groupName, Message message);

    void UnregisterGroup(string groupName);
}
```

Figura 5.3: Interface IBrokerClient

5.3 User

Para implementar o cliente, definiu-se a interface apresentada na figura 5.4.

```
public interface IUser
{
    string GetUserName();

    int GetUserNumber();

    void AcceptMessage(Message message);
}
```

Figura 5.4: Interface IUser

A implementação da interface também extende de *MarshalByRefObject* porque a referência do objeto será enviado para um *broker*.

5.4 Controlo de Concorrência

Tanto o *central manager* como o *broker* seguem o padrão *singleton*, como descrito anteriormente, e por essa razão o CLR do .NET irá atender os diferentes pedidos em diferentes *threads*. Como as *threads* irão aceder à mesma instância, é necessário garantir controlo de concorrência. Para garantir o acesso concorrente ao estado dessas instâncias, foi usado a biblioteca *Concurrent* do .NET.

5.5 Ficheiros de Configuração

De forma a não comprometer o código com os URLs, protocolos, implementações e canais, foi usado ficheiros de configuração. Para poder ter vários URLs para o mesmo tipo, foi usada

a *tag appSettings*. Desta forma, é possível haver vários URIs para o mesmo tipo. Cada URI está associado a um *broker*. O *type* e o *assembly* são duas entradas na *tag* usadas para construir instâncias da classe *WellKnownClientTypeEntry*. Assim, tem-se definido estáticamente quais os *brokers* existentes neste sistema.

O *TypeFilterLevel* usado em todos os ficheiros é *Full* porque todas as interações entre as partes recebem como argumento um objeto que estende de *MarshalByRefObject*.

Capítulo 6

Tolerância a Falhas

Com esta arquitetura existem cenários em que o servidor central possa falhar e as trocas de mensagens entre regiões não aconteça, mas que dentro de cada região continue a funcionar. Se o servidor central voltar a conectar-se, irá realizar uma nova conexão com os servidores regionais que conhece. Esta solução permite a troca de mensagens entre utilizadores dentro da mesma região enquanto não estiver conectado um servidor central.

Um outro cenário seria um servidor regional a falhar. Neste caso só a região desse servidor é que não conseguiria trocar mensagens. Mensagens de outras regiões também não chegariam à região afetada. Caso o servidor central envie mensagens para este servidor regional, tal não é possível, pois este servidor foi desconectado. Esta solução tem um inconveniente. O servidor central tem guardado informação de servidores regionais que podem já não estar conectados. O servidor central não tem maneira de identificar se o servidor regional foi desconectado ou se é um problema de comunicação. Se um servidor regional falhar e ficar ativo novamente, irá conseguir conectar-se com o servidor central. O problema desta solução é perder a informação dos utilizadores e dos grupos existentes.

Se um cliente desconectar-se sem informar o servidor regional, os dados ficam guardados nesse servidor, pois não há forma de saber que o cliente tem problemas de comunicação ou se está mesmo desconectado. Uma solução alternativa passaria por contar o número de chamadas consecutivas que o servidor regional faz ao cliente e este não responde com sucesso. Teria de ser atribuído um número máximo de chamadas consecutivas falhadas, e eliminar a informação deste cliente no servidor regional após o número de chamadas ultrapassar o limite. O problema com esta solução é que o cliente poderia estar novamente disponível após o limite de tentativas, e assim o servidor regional iria apagar informação de um cliente ativo.

Capítulo 7

Manual de utilização

Para usar o sistema desenvolvido, terá de executar os seguintes passos dentro da pasta Install:

1. Lançar o executável *CentralManagerServer.exe* que consta dentro da pasta Central Manager;
2. Dentro da pasta Brokers, existem duas pastas. Em cada uma delas, deverá lançar o executável *BrokerServer.exe*;
3. Para executar um ou mais clientes, deverá lançar o executável *UserFormImpl.exe* presente na pasta User, consoante o número de utilizadores desejado.

7.1 Utilização do Cliente

Na figura 7.1 é apresentada a interface gráfica do utilizador. Relativamente a esta figura, existem retângulos numerados. Cada retângulo é explicado de seguida.

No retângulo 1, pode selecionar o *broker* à qual o utilizador pretende-se registar. Para tal, necessita de indicar o seu número e o seu nome. Para se registar, deve preencher o formulário e carregar no botão *Register*. Para fazer *Unregister*, deve carregar no botão *Unregister*.

No retângulo 2, existe a possibilidade de criar ou remover grupos. Para criar ou remover um grupo, basta indicar o seu nome e carregar no botão *Add* ou *Remove* para adicionar ou remover um grupo, respetivamente.

No retângulo 3, pode adicionar um utilizador a um dado grupo à qual pertence. Para realizar esta operação, deverá indicar o número do utilizador que pretende adicionar e o nome do grupo ao qual irá adicionar. Havendo preenchido o formulário, deverá carregar no botão *Add*.

O retângulo 4, permite enviar mensagens para um utilizador ou para um grupo. Para enviar uma mensagem para um utilizador, deverá selecionar o *radio button User* e indicar o número do utilizador à qual pretende enviar a mensagem. A caixa de texto *Send a message* permite escrever a mensagem que pretende enviar. Para enviar, basta pressionar o botão do teclado *Enter*. Se pretender enviar a mensagem para um grupo, o procedimento repete-se, diferindo apenas na escolha do *radio button*, que deverá ser *Group*.

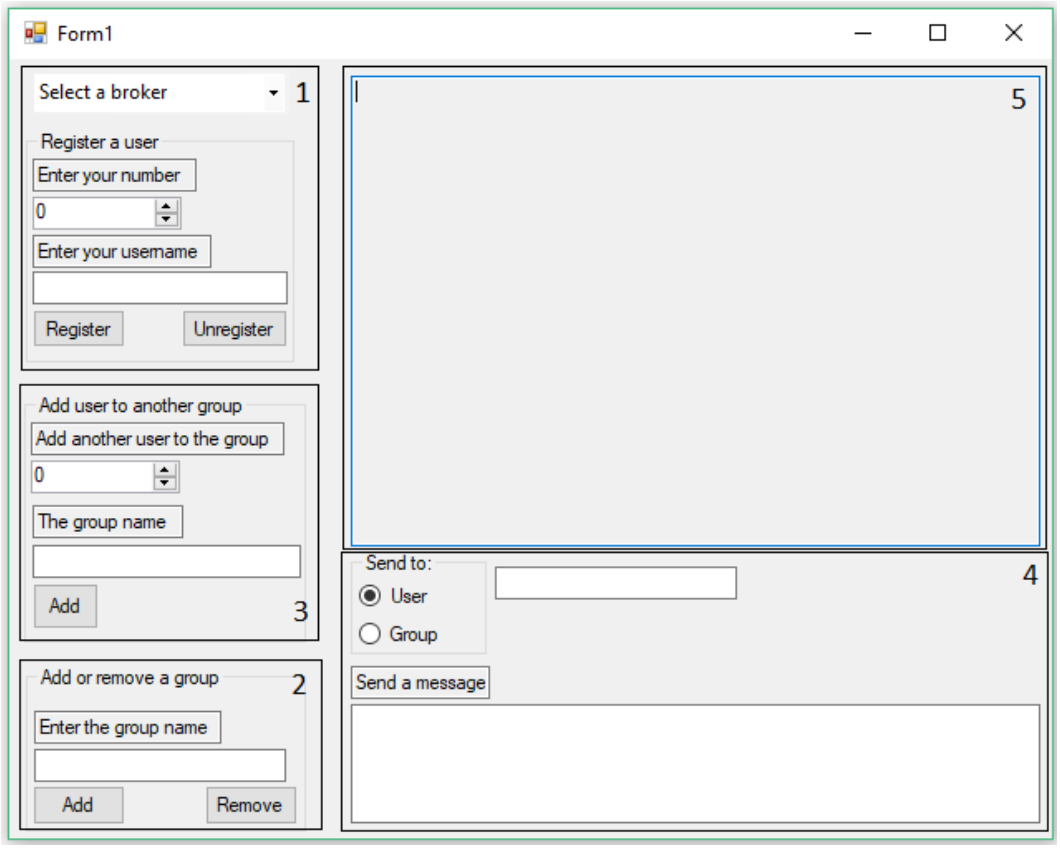


Figura 7.1: User Form

O retângulo 5 é onde se apresenta as mensagens enviadas e recebidas, quer por utilizadores, quer por grupos. As mensagens vêm identificadas pelo nome do utilizador caso exista. Caso contrário, são identificadas pelo número do utilizador.

Capítulo 8

Conclusão

Com o término do trabalho, foi possível aferir os vários conhecimentos relacionados com os objetos distribuídos da plataforma .NET. As dificuldades encontradas estiveram relacionadas com a definição da interação entre as partes, pensar nas possíveis falhas e tentar arranjar uma melhor forma de garantir tolerância a falhas. Foi possível consciencializar os problemas encontrados ao desenvolver um sistema distribuído, tais como tratamento de falhas, concorrência e escalabilidade. A apreciação final do trabalho desenvolvido é satisfatória.