

Fundamentos de Sistemas Distribuídos

Ano letivo 2022/2023

Enunciado do Trabalho Prático de Fundamentos de Sistemas Distribuídos (Fase2)

28 de outubro de 2022 (v1.0)

Introdução

Este documento descreve o projeto a realizar no âmbito da unidade curricular de Fundamentos de Sistemas Distribuídos. A realização deste projeto pretende promover a aquisição de competências de programação distribuída, tentando assim corresponder a alguns dos objetivos de aprendizagem da unidade curricular. Este projeto será também a principal forma de avaliação dessas mesmas competências e um contexto de aprendizagem capaz de promover a consolidação dos conceitos mais teóricos abordados nas aulas. É um pressuposto deste enunciado que os alunos disponham de competências adequadas de programação em JAVA e trabalho em equipa.

Descrição geral do trabalho prático

O objetivo deste trabalho é desenvolver um sistema básico de partilha de pequenas mensagens de texto, designadas **posts**. O paradigma base do sistema é o **cliente-servidor**. Cada utilizador é representado por um programa, designado **agente do utilizador**, que atua como cliente de **servidor comum** a todos os agentes de utilizador. Para utilizar o sistema, o utilizador executa um agente de utilizador e este irá ligar-se ao servidor do sistema, passando a fazer parte de um chat público, no qual os utilizadores partilham os seus posts.

O servidor tem **duas funções fundamentais**: a primeira é manter a informação de sessão, ou seja, uma lista com agentes de utilizador que se encontram ligados num determinado momento e com os posts recentes; a segunda é receber os posts enviados por cada agente de utilizador e torná-los disponíveis para todos os outros utilizadores presentes na sessão.

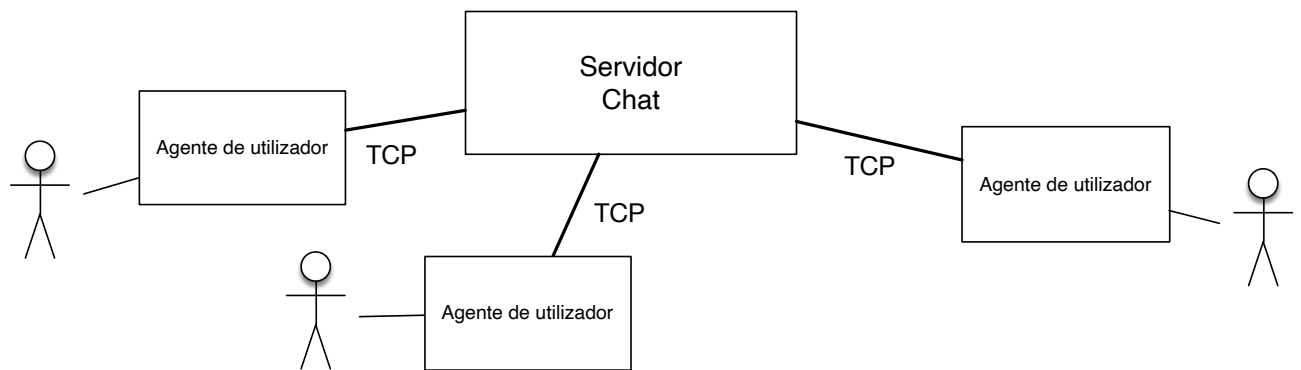
A partir da Fase 2, o sistema deve também permitir que os agentes de utilizador permitam a troca direta de mensagens entre utilizador com base numa interface Java RMI.

Especificação do sistema

Esta secção descreve a especificação geral do sistema. As especificações das fases anteriores mantêm-se inalteradas, sendo, no entanto, complementadas com as novas especificações introduzidas em cada uma das fases seguintes. Neste documento constam as especificações iniciais e as especificações complementares correspondentes à Fase 2 de desenvolvimento do projeto.

Componentes do sistema

O sistema é composto por um servidor e múltiplos agentes de utilizador que funcionam como clientes dos serviços proporcionados pelo servidor. Toda a comunicação entre o servidor e os agentes de utilizador é feita por troca de mensagens através de sockets TCP.



Cada agente de utilizador representa um utilizador do sistema. O utilizador inicia o agente, indicando o servidor a que se pretende ligar. Após o estabelecimento da ligação, o utilizador passa a fazer parte de uma sessão de chat na qual estão presentes todos os utilizadores que se ligaram ao mesmo servidor. Quando um utilizador coloca um post no chat, esse post é apresentado a todos os outros participantes da sessão. Para que isso aconteça, o utilizador interage com o seu agente de utilizador para escrever o texto do post e o agente envia para o servidor uma mensagem com esse texto. Esse post é acrescentado ao chat público e à medida que os vários clientes forem fazendo pedidos de atualização ao servidor, vai-se difundindo por todos os clientes.

Sessão

Antes de poderem fazer parte do chat, os agentes de utilizador precisam de iniciar uma sessão junto do respetivo servidor. Para isso devem enviar ao servidor uma mensagem do tipo `SESSION_UPDATE_REQUEST`, que serve para um agente de utilizador se sincronizar com a sessão do servidor. A mensagem tem de incluir o *nickname* do respetivo utilizador (informação obrigatória) que funcionará como um identificador único do utilizador. Além disso, poderá também incluir outros campos de informação que descrevam melhor o utilizador ou as suas preferências para participação na sessão. Ao receber um `SESSION_UPDATE_REQUEST`, o servidor deve acrescentar o novo cliente à lista de presenças e guardar a respetiva informação.

As mensagens do tipo `SESSION_UPDATE_REQUEST` servem, portanto, para o agente de utilizador iniciar a sessão quando se liga pela primeira vez ao servidor, mas será também o tipo de mensagem que o agente de utilizador deverá enviar periodicamente ao servidor para manter ativa a sua presença na sessão e para atualizar a sua informação sobre o estado da sessão (ou seja quem está ligado e quais os posts mais recentes). Assim, sempre que o servidor recebe uma mensagem do tipo `SESSION_UPDATE_REQUEST`, quer seja a inicial ou uma subsequente, o servidor responde com uma mensagem do tipo `SESSION_UPDATE`. Esta mensagem deve incluir a listagem dos agentes com sessões ativas e a listagem dos últimos 10 posts recebidos pelo servidor.

O servidor não mantém qualquer informação de estado sobre os clientes, apenas a sua visão centralizada do estado do sistema. Não tem, portanto, qualquer comportamento específico para qualquer um dos clientes e não guarda qualquer informação sobre que mensagens enviaram antes ou o que é que já sabem sobre a presença de outros clientes ou sobre as mensagens anteriormente trocadas entre eles. A mensagem `SESSION_UPDATE` é, portanto, criada com base apenas na sua própria informação de sessões abertas e na lista de posts realizados até ao momento por todos os agentes, sem quaisquer considerações sobre o que cada um deles já possa ou não saber sobre as alterações recentes. Assim, a mensagem `SESSION_UPDATE` deve ser exatamente a mesma para todos os agentes (mesmo para o que poderá ter

desencadeado a alteração de estado mais recente) e consiste sempre na lista atualizada de presenças na sessão e os últimos 10 posts enviados para o servidor.

Se o servidor estiver sem receber nenhuma mensagem de um agente de utilizador mais do que o número de segundos estipulado no parâmetro `SESSION_TIMEOUT`, a respetiva sessão é terminada e a ligação via sockets é fechada. Um agente será considerado em sessão sempre que tenha enviado uma mensagem (de qualquer tipo) para o servidor dentro do limite definido pelo `SESSION_TIMEOUT`. O valor, em segundos, sugerido para esse parâmetro é de 120.

Publicação de posts

Quando um utilizador escreve um post no seu agente de utilizador, o que pressupõe que o mesmo já faça parte de uma sessão ativa, isso dá início ao processo de envio de uma mensagem do tipo `AGENT_POST`. O agente de utilizador envia para o servidor uma mensagem deste tipo contendo o conteúdo do post e possivelmente outras informações complementares sobre o mesmo. Ao receber uma mensagem `AGENT_POST` válida, o servidor atualiza a sua informação interna, acrescentando ao chat o post que acabou de receber.

Difusão de mensagens pelo servidor

Os grupos que tenham concluído com sucesso os procedimentos já apresentados, podem incluir como funcionalidade adicional a possibilidade de ser o servidor a tomar a iniciativa de notificar todos os clientes sempre que existe alguma alteração no estado do sistema. Ou seja, quando há um novo membro ou um novo post, o servidor toma ele próprio a iniciativa de enviar uma mensagem do tipo `SESSION_UPDATE` para cada um dos agentes com sessão ativa. Desta forma-se consegue-se que as atualizações sejam propagadas mais rapidamente e evita-se que os clientes tenham de estar constantemente a contactar o servidor só para saberem se existe algo de novo.

Fase 2: Troca direta de mensagens entre agentes de utilizador com base em Java RMI

Na fase 2 de desenvolvimento do projeto pretende-se permitir que, para além de todas as funcionalidades já anteriormente descritas, seja também possível aos agentes de utilizador enviarem mensagens privadas diretamente uns para os outros. Estas mensagens seguem um paradigma peer-to-peer, uma vez que são feitas diretamente entre os agentes de utilizador, sem qualquer intervenção ou conhecimento do servidor. É, ainda assim, um modelo peer-to-peer híbrido porque o servidor é usado pelos agentes para tomarem conhecimento da presença uns dos outros e obter os respetivos pontos de contacto.

Esta nova funcionalidade será implementada em Java RMI, pelo que cada agente de utilizador deverá implementar uma interface remota a partir do qual os outros agentes lhes possam enviar as mensagens diretas. A introdução desta possibilidade adicional implicar acrescentar as seguintes funcionalidades:

1. Os agentes de utilizador com suporte RMI devem disponibilizar uma interface RMI através da qual os outros agentes de utilizador possam submeter mensagens. Quando uma dessas mensagens é recebida, a mesma deverá ser imediatamente apresentada ao utilizador com a indicação de que se trata de uma mensagem privada e de quem a enviou.
2. Para poder receber este tipo de mensagens, um agente de utilizador precisa de criar um objeto remoto Java que implemente uma interface RMI bem conhecida e que é comum a todos os agentes. Essa interface tem apenas um método que possibilita o envio da mensagem, e cuja especificação é a seguinte:

```
import java.rmi.Remote;  
import java.rmi.RemoteException;
```

```
public interface PrivateMessaging extends Remote
{
    String sendMessage(String name, String message) throws RemoteException;
}
```

O parâmetro **name** corresponde ao nome do utilizador que enviou a mensagem. O parâmetro **message** corresponde ao texto da mensagem. O método deve retornar como resultado o nome do utilizador que recebeu a mensagem.

3. Após a criação do objeto remoto, e para que este possa ser utilizado remotamente, o mesmo deve ser registado num servidor RMI Registry localizado na mesma máquina onde está a correr o agente de utilizador. O serviço deve ser registado com o nome `PrivateMessaging`.
4. Os agentes de utilizador com este suporte RMI devem incluir na informação sobre si próprios que enviam para o servidor, as informações adicionais referentes ao RMI. Essa informação deverá passar a ser incluída pelo servidor nas mensagens `SESSION_UPDATE` que o servidor usa para enviar para os agentes de utilizador a informação mais recente sobre quem está presente na sessão e sobre os posts mais recentes. Assim, e para cada utilizador presente na sessão, os agentes de utilizador irão receber o nome do utilizador, mas também o respetivo endereço IP, e a porta do *RMI registry*, se diferente da porta 1099 (default do RMI).

Os grupos que tenham concluído com sucesso os procedimentos já apresentados, podem desenvolver um interface novo em que o método `sendMessage` inclua um parâmetro adicional correspondente ao endereço remoto de quem enviou a mensagem. Essa referência para o objeto RMI do agente que efetuou o pedido irá permitir que o agente que recebeu a mensagem possa responder imediatamente, sem ter de recorrer ao Registry para obter essa mesma referência.

Fase 3: Segurança

Na fase 3, o sistema será melhorado através da introdução de mecanismos de segurança que permitam colmatar as principais vulnerabilidades da atual implementação. Mais especificamente, pretende-se que o sistema passe a garantir a confidencialidade das mensagens trocadas e a autenticação dos vários elementos que o compõem.

Trabalho a realizar

O projeto está organizado em 3 fases. Todas as fases serão desenvolvidas em Java e estão desenhadas de modo a serem evoluções sucessivas do mesmo sistema. Este enunciado refere-se apenas às fases 1 e 2 de execução do projeto. Posteriormente, será disponibilizado o enunciado referente à fase 3.

Adicionalmente ao trabalho já desenvolvido na primeira fase, e perfeitamente integrado no mesmo, cada grupo deverá agora criar novas versões de agente de utilizador que integrem todas as funcionalidades anteriores mais as funcionalidades aqui apresentadas. A demonstração final deverá apresentar três ou mais agentes a ligarem-se ao servidor, a atualizarem a lista de presenças sempre que um utilizador inicia ou termina uma sessão e a difundir entre todos os posts que vão sendo publicados em cada um dos agentes de utilizador. Deverá também mostrar um qualquer desses agentes a enviar diretamente uma mensagem privada para outro e a mesma a ser de imediato apresentada por esse outro agente ao seu respetivo utilizador.

A interação com o utilizador poderá ser baseada num interface gráfico ou texto. O agente de utilizador deve permitir indicar um endereço do servidor e uma porta de ligação ao servidor, devendo, no entanto, existir valores pré-definidos para ambos os parâmetros.

Do mesmo modo, a execução do servidor deve poder permitir especificar a respetiva porta de operação e o valor do parâmetro `SESSION_TIMEOUT`, sendo que também neste caso devem existir valores pré-definidos.

Questões Fase 2

Sendo este um projeto prático com objetivos pedagógicos, deve também servir de enquadramento para os conceitos teóricos da unidade curricular. Nesse sentido, cada grupo deverá incluir no seu relatório a resposta às seguintes questões:

Q1: Especificações abertas

Imagine que se pretendia que o vosso sistema fosse um sistema aberto em que múltiplas entidades pudessem desenvolver os seus próprios agentes de utilizador, incluindo as funcionalidades RMI. Focando-se apenas nas alterações introduzidas nesta fase 2, indique que especificações seriam necessárias para que alguém que não tivesse qualquer outro conhecimento sobre a vossa implementação para além do que é dito neste enunciado, pudesse desenvolver autonomamente um novo agente de utilizador capaz de interagir com o vosso agente de utilizador para o envio de mensagens privadas.

Q2: Difusão de mensagens pelo servidor

Na fase 1 foi sugerido que o servidor poderia ter a iniciativa de notificar os agentes de utilizador sempre que ocorresse uma mudança no estado do sistema, tomando ele próprio a iniciativa de enviar uma mensagem de `SESSION_UPDATE` para todos os agentes com sessões ativas. Sem entrar em detalhes de programação, explique de que modo se poderia tirar proveito do RMI para fazer algo equivalente.

Q3: Alternativa com sockets.

Imagine que se pretendia implementar esta funcionalidade de mensagens diretas, sem utilizar o RMI e recorrendo antes à troca de mensagens por sockets. De forma muito resumida, indique quais seriam os principais elementos de uma solução desse tipo.

Instruções para a execução do trabalho

Este enunciado pode vir a ser alvo de alguma correção ou refinamento, pelo que se sugere que seja sempre utilizada a última versão que será sempre a que consta da página de e-learning da UC. A versão dos documentos está indicada no cabeçalho deste documento

Grupos de trabalho

O trabalho será realizado em grupo de 3 elementos nas aulas presenciais, sendo complementado por trabalho realizado não presencialmente.

Entregas do projeto

As entregas do projeto irão desenrolar-se em linha com o final de cada uma das fases de execução, existindo, portanto, três entregas. As datas concretas para entrega das fases seguintes serão anunciadas nos respetivos enunciados

Prazo de entrega para a fase 1: 23:59 do dia 31/10/2022

Prazo de entrega para a fase 2: 23:59 do dia 30/11/2022

Praxo de entrega para a fase 3: 23:59 do dia 3/1/2023

Apresentações finais: A agendar a partir de 4/1/2023

A entregas do código e do relatório deverão ser realizadas em formato eletrónico na página de elearning da UC, usando o respetivo link SafeAssign. Essa entrega será composta pelos seguintes elementos:

1. **Relatório do código:** Este elemento corresponde a um único **ficheiro em formato .pdf** correspondente à concatenação dos vários ficheiros `.java` que foram desenvolvidos.
2. **Código desenvolvido:** Este elemento corresponde a um ficheiro em formato zip com todo os ficheiros de código desenvolvidos e prontos a executar. Apenas devem ser incluídos ficheiros `.java`.
3. **Relatório do projeto:** O relatório do projeto deve ter no máximo 6 páginas, abrangendo 3 secções. A primeira secção é apenas à página de rosto do relatório, e deve conter a identificação do grupo e dos seus elementos, incluindo as respetivas fotos. A segunda secção é referente à implementação e deverá ter entre duas a três páginas. Esta secção deve descrever de forma sucinta, mas rigorosa, a implementação realizada. O objetivo é fazer a ponte entre a especificação apresentada neste enunciado (sem a repetir) e os detalhes da implementação realizada, como por exemplo que classes são usadas em cada componente e com que função. A terceira secção deverá ter também entre duas a três páginas e deverá dar resposta às questões específicas colocadas neste enunciado.

Autoria

A autoria do projeto será avaliada com recurso à ferramenta *SafeAssign* do sistema *Blackboard* ou equivalente. Uma vez que todos os grupos estão a fazer o mesmo projeto é normal que ocorra alguma partilha de conhecimento entre os grupos. No entanto, quaisquer interações entre grupos deverão ter sempre em conta o princípio fundamental de que cada grupo tem de chegar de forma autónoma e independente à sua solução para o problema. Para evitar problemas com a autoria dos trabalhos, sugere-se que os alunos tenham em conta os seguintes referenciais relativamente ao que é ou não permitido:

- Não prestar nem receber qualquer ajuda que seja específica do problema proposto, como por exemplo a forma de implementar determinada parte do trabalho.
- **NUNCA partilhar código ou relatório** entre os grupos, já que trabalhos que apresentem semelhanças de implementação ou no relatório serão encarados como uma tentativa de fraude e serão anulados.
- Os grupos podem discutir entre si:
 - Os objetivos e a interpretação do problema.
 - A utilização em geral (não especificamente no contexto deste trabalho) das tecnologias necessárias para a realização do projeto.
- Qualquer utilização de código retirado de outras fontes deve ser explicitamente assinalada e justificada.
-

Avaliação

A avaliação final terá em conta o trabalho realizado em cada momento de avaliação, o código final, o relatório final, o cumprimento dos prazos e a defesa/demonstração do trabalho realizada individualmente por cada aluno, contribuindo cada um destes elementos de igual forma para a classificação de cada momento de avaliação do projeto. Nos casos em que a defesa individual não seja satisfatória (o que implica a atribuição de uma nota inferior em relação à classificação do trabalho, que seja correspondente ao contributo/demonstração, ou até à reprovação do aluno), e se o aluno o pretender, haverá uma segunda chamada de defesa do contributo para o projeto (a agendar conforme disponibilidade do aluno e equipa docente).

Guião de avaliação para a fase 1

Lista de elementos a avaliar na entrega do projeto, dos mais básicos para os mais sofisticados.

1. O agente de utilizador consegue estabelecer uma ligação ao servidor, localizado numa outra máquina, e enviar corretamente uma mensagem `SESSION_UPDATE_REQUEST` para dar início a uma sessão.
2. Ao receber uma mensagem `SESSION_UPDATE_REQUEST`, o servidor consegue responder com uma mensagem `SESSION_UPDATE` contendo a atual informação de estado (sessões+posts recentes).
3. Sempre que um agente de utilizador recebe uma mensagem privada, a mesma é imediatamente apresentada ao utilizador com a indicação de que se trata de uma mensagem privada e de quem a enviou.
4. Ao receber uma mensagem `SESSION_UPDATE`, o cliente consegue processar adequadamente a informação recebida e apresentá-la ao utilizador.
5. Quando já existe uma sessão com o servidor, o agente de utilizador permite que este possa escrever um post, sendo este imediatamente enviado para o servidor através de uma mensagem `AGENT_POST`. O servidor deve registar esse novo post e o mesmo deverá passar a ser incluído nas próximas mensagens de `SESSION_UPDATE` enviadas pelo servidor.
6. O servidor suporta múltiplas ligações simultâneas, podendo todos os procedimentos anteriores ser executados concorrentemente por vários agentes de utilizador, sem pôr em causa o correto funcionamento do sistema.
7. O agente de utilizador envia regularmente para o servidor, com uma periodicidade configurável, uma mensagem de `SESSION_UPDATE_REQUEST`, de modo a manter razoavelmente atualizada a informação que apresenta ao utilizador sobre o estado do sistema.
8. Tanto o agente de utilizador como o servidor podem ser executados com parâmetros opcionais que, quando usados, se sobrepõem a valores pré-definidos para esses mesmos parâmetros.
9. O agente de utilizador e o servidor fazem uma gestão adequada da comunicação de modo a manter ativa a sessão. Esta gestão deve ter em conta a regularidade dos pedidos de `SESSION_UPDATE_REQUEST` enviados pelo agente de utilizador, o parâmetro `SESSION_TIMEOUT`, e as configurações que permitem manter os sockets abertos, mesmo quando, durante algum tempo, não existe troca de mensagens entre as partes.
10. Quando recebe uma mensagem que atualiza o seu estado interno, o servidor toma a iniciativa de enviar uma mensagem do tipo `SESSION_UPDATE` para todos os agentes de utilizador com sessão aberta. Os agentes de utilizador podem continuar a enviar mensagens periódicas de `SESSION_UPDATE_REQUEST` mas apenas para manter a sessão aberta, pois deixa de ser necessário fazê-lo para receber atualizações sobre o estado do sistema.

Guião de avaliação para a fase 2

Lista de elementos a avaliar nesta entrega do projeto, dos mais básicos para os mais sofisticados.

1. O agente de utilizador consegue criar o objeto remoto, registá-lo num RMI registry local e enviar essas indicações para o servidor.

2. A informação de sessão enviado pelo servidor nas mensagens SESSION_UPDATE inclui a indicação de quais os agentes que podem receber mensagens diretas por RMI, bem como quaisquer dados adicionais que sejam necessários para o estabelecimento dessa comunicação
3. O agente de utilizador ao apresentar a informação de sessão indica explicitamente quais os agentes com suporte para mensagens diretas por RMI e permite ao utilizador seleccionar um desses agente como destinatário.
4. No agente de utilizador pode-se seleccionar um destinatário localizado numa outra máquina, compor uma mensagem privada e solicitar o seu envio, sendo esta enviada por RMI.
5. Os agentes de utilizador conseguem trocar diretamente as suas referências remotas entre si, evitando assim a necessidade de recorrer sempre ao RMIRegistry. (Extra)