Mestrado Integrado/Licenciatura em Engenharia e Gestão de Sistemas de Informação

Fundamentos de Sistemas Distribuídos

Ano letivo 2022/2023

Enunciado do Trabalho Prático de Fundamentos de Sistemas Distribuídos

4 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

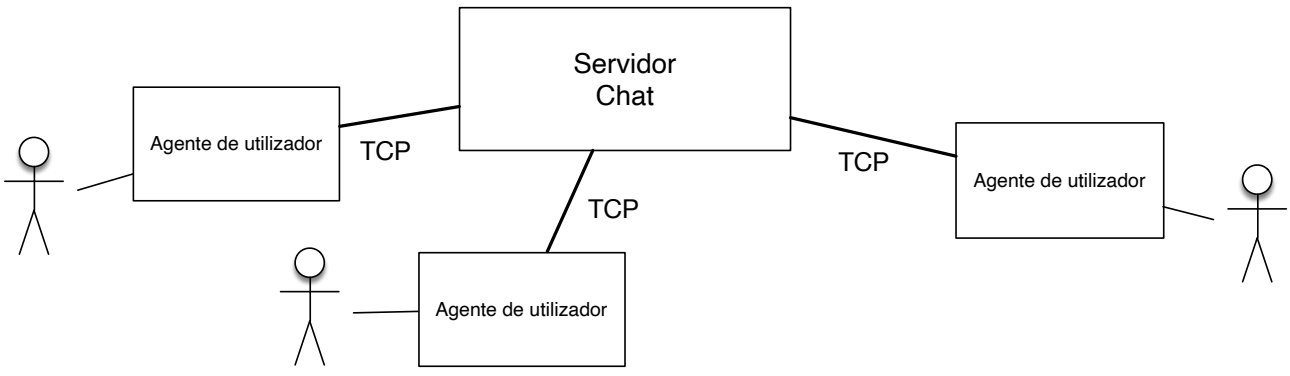
O objetivo deste trabalho é desenvolver um sistema básico de partilha de pequenas mensagens de texto, designadas **posts**. O paradigma base do sistema é o **cliente-servidor**. Cada utilizador é representado por um programa, designado **agente do utilizador**, que atua como cliente de **servidor comum** 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.

Especificação do sistema

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 mantem 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.

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 à fase 1 de execução do projeto, uma fase centrada na comunicação por sockets entre os agentes de utilizador e o servidor do sistema. Posteriormente, serão lançados os enunciados referentes às fases 2 e 3 de execução do projeto.

A primeira fase corresponde à implementação da especificação do sistema descrita neste enunciado. Cada grupo deverá desenvolver uma implementação de um servidor e pelo menos uma implementação de um agente de utilizador. 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.

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.

Fase 2: RMI

Para além da troca de posts entre todos os clientes, a fase 2 irá permitir também que os clientes possam trocar entre si mensagens diretas. 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 um interface remoto a partir do qual os outros agentes lhes possam enviar as mensagens diretas.

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.

Questões

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. Sem repetir o que já é dito no enunciado, acrescentem todos os detalhes que considerem necessários para que alguém que não tivesse qualquer outro conhecimento sobre a vossa implementação, pudesse desenvolver autonomamente um novo agente de utilizador capaz de interagir com o vosso servidor, da mesma forma que o que foi desenvolvido no contexto deste projeto.

Q2: Falha de componentes

Num sistema distribuído, qualquer um dos componentes pode a qualquer momento falhar de forma autónoma. Indique de que modo o vosso sistema está ou não preparada para lidar com situações de comportamento incorreto ou bloqueio por parte dos vários componentes que o compõem.

Q3: Limitações do sistema

O sistema descrito no enunciado é naturalmente muito simplista, o que facilita a sua implementação, mas pode criar obstáculos à implementação de determinadas funcionalidades ou à sua capacidade para suportar adequadamente requisitos não-funcionais como desempenho, resiliência ou capacidade de escalar para números muito grandes de sessões ou de mensagens. Identifique algumas das limitações concretas e indique a que opções de desenho do sistema é que poderão estar associadas.

Q4: Difusão de mensagens pelo servidor

Que fatores poderiam fazer com que a solução mais adequada passe por serem os clientes a tomar regularmente a iniciativa de contactar o servidor para saber se existe nova informação, ou, pelo contrário ser o servidor a tomar a iniciativa de notificar todos os clientes sempre que ocorra uma alteração?

Q5: Implementação em sockets UDP

Acha que este projeto poderia ser implementado com base em sockets UDP em vez de TCP? Quais serão neste caso as vantagens da utilização do TCP. Que fatores poderiam fazer com que uma solução UDP fosse mais adequada?

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. A constituição de cada grupo deverá ser comunicada ao docente do respetivo turno prático, durante a aula prática da semana de 3 a 7 de outubro.

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

O prazo de entrega para esta fase será às 23:59 do dia 31/10/2022

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 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. Ao receber uma mensagem SESSION_UPDATE, o cliente consegue processar adequadamente a informação recebida e apresentá-la ao utilizador.
4. Quanto já existe uma sessão com o servidor, o agente de utilizador permite que este possa escrever um post, sendo este de imediato 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 envidas pelo servidor.
5. 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.
6. O agente de utilizar 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.
7. 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.
8. 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.
9. 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.