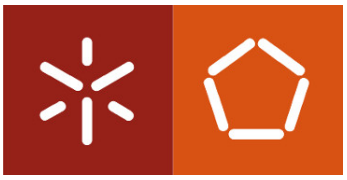


Universidade do Minho

Departamento de Informática

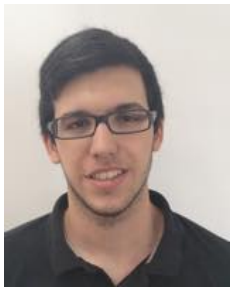
Sistemas Distribuídos

Trabalho Prático

Grupo Nº 20

Pedro Pereira (A80627)
Sofia Marques (A87963)
Pedro Pereira (A89232)
José Martins (A90122)

25 de janeiro de 2021



Pedro Pereira A80627

Sofia Marques A87963

Pedro Pereira A89232

José Martins A90122

Introdução

Sistemas distribuídos lidam normalmente com vários utilizadores e, como tal, devem estar preparadas para responder a vários pedidos feitos simultaneamente. Assim, neste tipo de sistemas, o controlo de concorrência torna-se essencial para que haja um bom desempenho, mantendo ao mesmo tempo a consistência dos dados.

Neste trabalho procuramos explorar as funcionalidades do modelo cliente-servidor com comunicação orientada à conexão via TCP, através do desenvolvimento de um servidor multithreaded que funciona como intermediário na comunicação entre clientes, utilizando a linguagem de programação Java para implementar tanto o servidor como o cliente.

O tema desenvolvido consiste na implementação de uma plataforma inspirada no problema do rastreio de contactos e deteção de concentração de pessoas.

O cliente tem a possibilidade de se registar e fazer o respetivo login na aplicação. Cabe a este manter a aplicação atualizada quanto à sua localização, de forma a que seja possível identificar o número de pessoas num dado local, ou até mesmo saber quando uma localização se encontra vazia no intuito de se deslocar para lá. Espera-se ainda que o cliente comunique ao servidor se está infetado.

O servidor, por sua vez, deverá manter em memória a informação relevante em relação aos utilizadores da plataforma, de forma a gerir a mesma. É importante referir ainda que este é o responsável pelos avisos aos clientes aquando de uma localização vazia, e por alertar os clientes que tenham estado em contacto com um utilizador infetado.

Estrutura

TaggedConnection

Classe que contém o *Socket* que permitirá a comunicação entre cliente e servidor, os respetivos Data[Input|Output]Stream e dois locks. Esta classe implementa ainda métodos que permitem o envio e receção de tramas. Está ainda implementada uma classe estática que representa a trama a ser enviada entre o cliente e servidor.

Client

Classe responsável por oferecer uma interface com o utilizador, através da qual envia os inputs relativos a cada funcionalidade para o Servidor. Tem também a função de receber mensagens do Servidor, algumas das quais apresenta posteriormente ao utilizador.

Server

Contém as classes MainWorker, ServerSocketMainWorker, WaiterWorker, ServerSocketWaiterWorker, InfectedWaiterWorer e ServerSocketInfectedWaiterWorker. Reponsável pela receção e processamento dos dados recebidos de acordo com a funcionalidade escolhida. Cria os ServerSockets necessários de acordo com os pedidos recebidos e conexões com clientes recebidas.

ServerStatus

Classe que inclui um hashMap que relaciona os usernames dos utilizadores com as respetivas classes de UserStatus que lhe estão associados. Contém também o Mapa que irá ser modificado e consultado conforme as interações do cliente com o servidor. Esta classe é também responsável por suportar todas as alterações do estado do servidor que podem ser requisitadas pelo cliente.

Mapa

Classe que implementa o mapa, relacionando num HashMap as coordenadas com o respetivo CoordStatus, que contém a informação relevante a estas. Implementa ainda operações lógicas sobre o mesmo.

Coordinates

Classe que representa uma coordenada de um mapa, através de 2 inteiros, X e Y.

CoordStatus

Classe que inclui informação relativa a uma coordenada do mapa. Esta possui dois Sets que registam todos os utilizadores que se encontram nessa coordenada e todos os que por lá passaram. Contabiliza também o número de infetados que estiveram nessa coordenada bem como o número de *Thread*s que estão à espera de que essa localização esteja vazia. Contém ainda um lock para permitir controlo de concorrência e uma condição relacionada com esse lock.

UserStatus

Classe que contém a informação relativa a um utilizador, isto é, o seu username e password, um Set que contém todos os utilizadores com quem este já contactou (esteve na mesma coordenada ao mesmo tempo), diferentes booleanos que identificam se este utilizador está infetado, se teve contacto de risco e se possui autorização especial e guarda também o número de logins efetuados pelo utilizador.

Métodos de comunicação

Para fazer a comunicação entre cliente e servidor foi usado um protocolo de mensagens codificadas através das tags. Por exemplo, quando o utilizador quer fazer o login, envia ao servidor uma mensagem com tag 1 que indica ao servidor que a operação a tratar será de login. Este método foi utilizado ao longo do programa para o servidor perceber quais os pedidos do cliente.

É importante referir que nas funcionalidades de saber quando uma localização se encontra vazia e na receção de avisos de contacto de risco, é utilizada comunicação assíncrona pelo que

foram utilizados *Sockets* e ServerSockets com portas diferentes consoante o tipo de comunicação que se tratava (comunicação "normal", aviso de localização vazia ou aviso de contacto de risco) permitindo que o utilizador continue a utilizar a aplicação sem que esta fique bloqueada.

Controlo de concorrência

Durante a resolução do trabalho foram utilizadas algumas técnicas de controlo de concorrência para impedir que os resultados fossem alterados a meio de uma leitura, causando resultados errados. Estes são apenas necessários do lado do servidor, uma vez que será este que irá lidar com vários clientes simultaneamente e estará suscetível a corridas.

Na classe TaggedConnection foram utilizados dois locks, wlock e rlock, permitindo o envio e receção, respetivamente, de dados através da conexão estabelecida simultaneamente.

Na classe CoordStatus é utilizado um lock de maneira a controlar o acesso às operações que são efetuadas sobre os dados referentes a uma coordenada.

Na classe UserStatus é também utilizado um lock controlando o acesso aos dados e consequentemente as operações referentes ao estado de um utilizador.

A classe Mapa foi um ponto de maior debate em relação à implementação de controlo de concorrência, tendo por fim sido decidido que não seria necessário a utilização de locks. Isto devido ao facto da estrutura do HashMap ser constante, e as operações serem feitas sobre os CoordStatus contidos no HashMap, que já contêm controlo de concorrência. Desta forma será possível que o estado de várias coordenadas seja alterado simultaneamente.

Funcionalidades Obrigatórias

Autenticação e Registo

Estas funcionalidades permitem que um utilizador estabeleça uma conexão de forma a conseguir interagir com a aplicação. Esta conexão pode ser feita por um Login quando o utilizador já está registado, ou através de um registo caso contrário. Nos dois casos o Cliente pede ao utilizador o seu username e a sua password. No caso do registo é também perguntado ao utilizador se este tem autorização especial. O Cliente envia depois os inputs do utilizador para o Servidor e espera a sua resposta, que contém a validação (ou não-validação) da autenticação.

Atualização da Localização

Esta funcionalidade serve para que um utilizador informe o Servidor da sua mudança de localização, indicando as coordenadas do local para o qual se deslocou. O Cliente envia estas coordenadas para o Servidor que verifica se estas são coordenadas válidas, efetua a operação e envia ao Cliente uma mensagem de validação (ou não-validação) desta. Por via desta funcionalidade também vão sendo registados os contactos existentes entre os diferentes utilizadores da aplicação.

Consultar número de pessoas numa localização

Esta funcionalidade permite ao utilizador saber o número de pessoas numa dada localização. O Cliente recebe as coordenadas da localização para a qual o utilizador quer saber o número de pessoas presentes e envia-as para o Servidor. Ao longo do decorrer do programa, o mapa vai sendo atualizado conforme o número de pessoas que se deslocam pelo mesmo. Sendo assim, o Servidor ao receber este pedido e as respetivas coordenadas necessita apenas de verificar o número de pessoas que se encontram na coordenada indicada, e enviar esse número ao Cliente, que posteriormente apresenta ao utilizador.

Aviso de localização vazia

Esta funcionalidade permite ao utilizador receber uma notificação posterior quando a localização indicada se encontrar com 0 ocupantes. O Cliente cria uma *Thread* que irá enviar as coordenadas e esperar a resposta do Servidor. No lado do Servidor é criado uma *Thread* que irá aguardar numa Condition do CoordStatus da coordenada recebida. Este será acordado pela *Thread* "principal"de um cliente no Servidor, aquando da mudança de localização de um utilizador dessa localização.

Comunicar infeção

No momento em que um dado utilizador se encontra infetado, este tem o dever de o informar à plataforma. Assim sendo, o Cliente comunica ao Servidor o seu novo estado. O Servidor através do nome do utilizador altera o respetivo UserStatus para indicar que está infetado. Após essa alteração é atualizado o Mapa, de maneira a que as coordenadas em que o utilizador infetado já esteve contenham a informação que terá lá havido mais um utilizador infetado. Por fim verifica os contactos que o mesmo teve de forma a alertá-los quanto a uma possível transmissão. Isto torna-se possível pois cada utilizador tem no seu UserStatus um conjunto de todos os contactos que teve. Após a comunicação o utilizador fica incapacitado de interagir de novo com a plataforma, recebendo um aviso de que as funcionalidades estão bloqueadas, caso o tente fazer.

Funcionalidades Adicionais

Notificação contactos infetado

Esta funcionalidade é da responsabilidade do Servidor. Quando um utilizador completa um Login, o Cliente cria uma *Thread* que envia o username para o Servidor e que fica à espera de receber uma notificação por parte do mesmo, caso o utilizador tenha estado em contacto com um outro infetado. Do lado do Servidor existe também uma *Thread* que recebendo o username do utilizador, fica bloqueada numa condição do UserStatus. Esta *Thread* apenas é acordada aquando do anúncio de uma infeção no sistema, pelo que se o utilizador em causa tenha estado em contacto com o infetado esta *Thread* irá ser desbloqueada de forma a poder avisar o Cliente de que é um contacto de risco.

Descarregar mapa

Esta funcionalidade permite aos utilizadores com a devida permissão obter um mapa atualizado com o número de utilizadores distintos que já estiveram em cada coordenada e também o número de utilizadores infetados distintos que já estiveram em cada coordenada. O Cliente envia o pedido

ao Servidor, sendo que este irá primeiro de tudo verificar se o utilizador tem as devidas permissões para obter esta informação. Caso não tenha, envia uma mensagem de erro específica. Caso tenha irá obter toda a informação necessária e envia-la.

Conclusão

Durante o desenvolvimento deste trabalho conseguimos pôr em prática muito do que estudámos durante este semestre, em especial utilizar os conceitos já elaborados nas aulas teórico-práticas de uma forma direta e prática. Usando os mecanismos de controlo de concorrência da linguagem Java, *Threads* e *Sockets* TCP, tenho já tido trabalhado com estes no desenvolvimento das fichas práticas ao longo do ano, conseguimos criar uma aplicação distribuída capaz de garantir o funcionamento de uma plataforma de rastreio e deteção e concentração de pessoas.

O aspeto que se apresentou de maior dificuldade foi a utilização de comunicação assíncrona. Pensamos que o principal problema da implementação final do programa que desenvolvemos estará na conclusão de todos os *Threads* criados ao longo do decorrer do programa, que infelizmente não conseguimos aperfeiçoar. Apesar disto pensamos ter alcançado maioritariamente os objetivos propostos pelos docentes, e sentimos que o desenvolvimento deste projeto terá sido benéfico para o aproveitamento de todo o grupo.