Alarme Covid

Trabalho realizado por:

João Figueiredo Martins Peixe dos Santos

Francisco Alves Andrade

Luís Filipe Cruz Sobral

Paulo Silva Sousa

A89520 João Santos

A89474 Luís Sobral

A89465 Paulo Sousa

A89513 Francisco Andrade

Grupo 14
Projeto SD
2020/2021
Universidade do Minho

Índice

1 Introdução **1**

2 Diagrama de Classes **1**

3 Descrição das Classes Implementadas **2**
3.1 Relacionadas com o Cliente 2
3.2 Relacionadas com o Servidor 2
3.3 Exceptions 3

4 Descrição das Operações **4**

5 Conclusão **5**

1 Introdução

Neste trabalho, realizado no âmbito da disciplina de Sistemas Distribuídos, foi-nos proposto o desenvolvimento de uma plataforma de rastreio e deteção de concentrações de pessoas, sendo que todos os dados podiam ser livremente armazenados no servidor. Esta plataforma pretende guardar as localizações dos vários utilizadores, com o objetivo de calcular concentrações de pessoas e registar todas as pessoas com quem cada utilizador esteve em contacto, possibilitando assim a notificação de utilizadores que tenham estado em contacto com casos positivos.

Esta plataforma deverá suportar várias funcionalidades tais como a autenticação e registo de utilizadores, informar o utilizador sobre o número de pessoas numa dada localização e informa-lo quando uma localização estiver vazia e, relativamente ao servidor, este deverá guardar as localizações dos utilizadores, quais destes estão doentes e notificar os potencias contagiados. É possível, também, que um utilizador com autorização especial aceda a um mapa com informação de quantos utilizadores e quantos doentes visitaram cada localização. Este serviço deverá ser implementado em Java com recurso a sockets e threads.

2 Diagrama de Classes

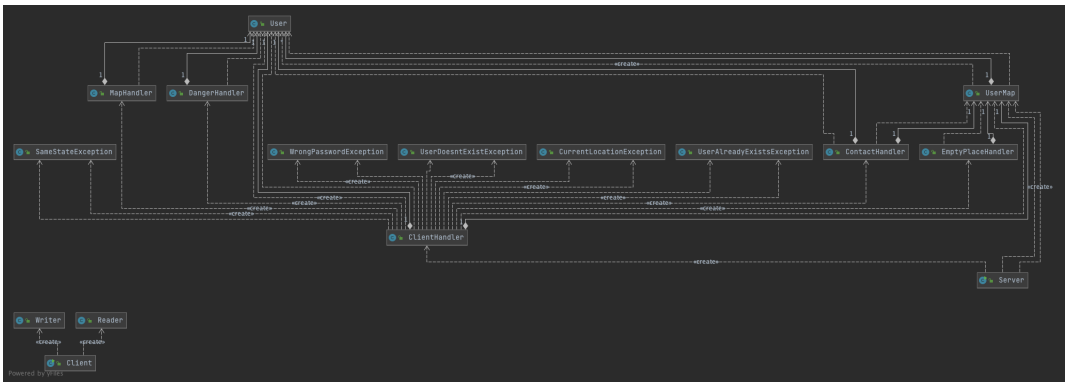

Figure 1: Diagrama de Classes

3 Descrição das Classes Implementadas

Neste projeto dividimos as classes em packages. Para isso, criamos um package para as classes relacionadas com o Cliente, um para as classes relacionadas com o Server e um Package para as Exceptions.

3.1 Relacionadas com o Cliente

- **Cliente**:

  Classe onde se efetua a ligação entre o cliente e o servidor e é responsável pela iniciação e execução de duas threads: uma associada a um objeto da classe Reader e outra associada a um objeto da classe Writer.

- **Reader**:

  Classe responsável pela leitura das respostas enviadas pelo servidor, imprimindo-as para o Stdout.

- **Writer**:

  Classe responsável por enviar para o servidor as mensagens correspondentes às opções escolhidas pelo utilizador.

3.2 Relacionadas com o Servidor

- **Server**:

  Classe responsável por criar uma ligação com cada utilizador, sendo responsável pela iniciação e execução de threads para comunicar com cada utilizador, estando estas associadas a um objeto da classe ClienteHandler.

- **User**:

  Classe responsável por armazenar os dados relativos ao User, tais como: username, password, se é um admin ou não, se está ou não online, se se encontra positivo na contração do vírus covid-19, as coordenadas identificadoras da localização dos utilizadores. Possui também um lock explícito para garantir a consistência dos dados apresentados.

- **UserMap**:

  Classe responsável por guardar e gerir um Map de Utilizadores.

- **MapHandler**:

  Classe responsável por guardar as informações nessessárias à funcionalidade de criar um mapa com todos os utilizadores e doentes que estiveram em cada localização.

- **ClientHandler**:

  Classe responsável por toda a comunicação com o cliente (responsavél por toda interface apresentada ao utilizador) e pela implementação de todas as funcionalidades do sistema. A cada conexão com um utilizador é associado um objeto desta classe.

- **ContactHandler**:

  Classe responsável pela atualização da lista de contactos de um utilizador quando o mesmo efetua o login ou quando altera a sua localização.

- **DangerHandler**:

  Classe responsavel por notificar os utilizadores que estiveram em contacto com utilizadores contagiados. O servidor cria, para cada utilizador, uma thread associada a um objeto desta classe que notifica o utilizador caso receba um signal a informar que esteve em contacto com um caso positivo.

- **EmptyPlaceHandler**:

  Classe responsável pela funcionalidade correspondente a informar o utilizador quando um local estiver vazio. Assim, caso o utilizador peça para ser informado sobre um local, o servidor cria uma thread associada a um objeto desta classe que notifica o utilizador caso receba um sinal a informar que a condição se verificou.

3.3 Exceptions

As Exceptions que criamos de modo a evitar erros e comportamentos inesperados foram as seguintes:

- CurrentLocationException
- SameStateException
- UserAlreadyExistsException
- UserDoesntExistException
- WrongPasswordException

4 Descrição das Operações

O servidor pode receber várias mensagens do utilizador, sendo que estas são recebidas e processadas pelo ClientHandler, que associa a mensagem recebida a uma funcionalidade do sistema. As várias funcionalidades do servidor serão explicitadas de seguida.

- **Login**:

  Esta operação corresponde à escolha da opção 1 do menu login, onde utilizador efetua o login no sevidor introduzindo o seu nome de utilizador e, de seguida, a sua password. É efetuada a verificação acedendo ao UserMap, utilizando para isso um ReadLock. Este processo é implementado pelo método interpreter_login.

- **Sair**:

  A opção de logout corresponde ao input 0 em todos os menus e resulta na finalização da conexão do utilizador com o servidor, alterando assim o valor de "online" na classe User para False.

- **Registar**:

  Esta operação corresponde à escolha da opção 2 do menu login, onde o novo utilizador introduz o seu novo nome de utilizador, a sua password e a sua localização (latitude e longitude). O nome de utilizador e a password são posteriormente verificados, acedendo ao UserMap, utilizando para isso um ReadLock. Este processo é implementado pelo método interpreter_register.

- **Atualizar Localização**:

  Esta operação corresponde à escolha da opção 1 do menu do utilizador ou do admin. Com esta funcionalidade o utilizador pode alterar a sua localização, sendo que apenas tem de inserir a localização para a qual se pretende mover. Caso a localização antiga fique vazia todos os utilizador à espera dessa informação são notificados. Além disso, é enviado um signal para todos os utilizadores na localização atualizarem o seu Set de contactos. Este processo é implementado pelo método interpreter_1.

- **Número de Pessoas numa dada Localização**:

  Esta operação, que corresponde à escolha da opção 2 do menu do utilizador ou do admin, permite ao utilizador informar-se sobre o número de pessoas que se encontram numa dada localização (inserida pelo utilizador). Este processo é implementado pelo método interpreter_2.

- **Pedir para informar sobre um Local**:

  A opção 3 do menu do utilizador ou admin está associada a esta operação. Esta funcionalidade permite ao utilizador receber uma notificação quando a localização escolhida se encontrar vazia. Para isso, esta cria uma thread que fica em espera até ser sinalizada de que a localização está vazia. Este processo é implementado pelo método interpreter_3.

- **Comunicar que está doente**:

  Esta operação corresponde à escolha da opção 4 do menu do utilizador ou do admin. O utilizador pode, com esta funcionalidade, informar o servidor que está doente. Assim, é enviado a todos os utilizadores que estiveram em contacto com este um signal, de modo a avisar que estiveram em contacto com um caso positivo. Este processo é implementado pelo método interpreter_4.

- **Mapa de localizações**:

  Esta operação apenas está disponivel para o admin e corresponde à opção 5. O admin pode com esta funcionalidade aceder a um mapa com informação de todos os utilizadores e doentes que estiveram em cada localização. Este processo é implementado pelo método interpreter_5, que cria, para cada utilizador, uma thread associada a um objeto da classe MapHandler que preenche o mapa com as localizações onde este utilizador esteve.

- **Comunicar que recuperou**:

  Esta operação apenas está disponivel para utilizadores com covid, onde podem avisar o servidor que recuperaram da doença e podem voltar a utilizar a aplicação com as outras funcionalidades. Este processo é implementado pelo método interpreter_4.

5 Conclusão

O presente trabalho prático permitiu a consolidação da matéria dada nas aulas teóricas, nomeadamente sobre as threads e sockets. Estudamos diferentes formas de implementação de um sistema distribuído, e o porquê das suas utilizações, entre outros aspetos.

Em suma, o grupo considera que este trabalho prático foi um bom meio de estudo e uma experiência positiva para a avaliação e consolidação de conhecimentos relativamente à UC de Sistemas Distribuídos.