Universidade do Minho

Departamento de Informática

Sistemas Distribuídos

Grupo nº 3

Carlos Preto (a89587)

Maria João Moreira (a89540)

Pedro Veloso (a89557)

Rui Fernandes (a89138)

1. Introdução

O presente trabalho foi realizado na Unidade Curricular de Sistemas Distribuídos e tem como objetivo o desenvolvimento de uma aplicação de rastreio de contactos próximos com uma pessoa infetada com o Covid-19.

O Covid-19, com origem em 2019, pode ser considerado um dos vírus mais contagioso da atualidade. Este vírus apresenta a particularidade de pessoas assintomáticas, serem igualmente contagiosas. Neste sentido, é fundamental registar os movimentos das pessoas, de modo a se poder identificar e notificar os contactos próximos, aquando de uma situação de um teste positivo.

Com vista a solucionar este problema, foi desenvolvida a aplicação "Alarme Covid" que, através da localização física de uma pessoa registada, permite avisar um utilizador sempre que este se encontrou num local com um infetado. Além disso, os utilizadores conseguem ter acesso ao número de pessoas presentes num espaço físico, de modo a evitar concentrações de multidões. Os utilizadores podem, inclusive, parametrizar a aplicação para receber uma notificação sempre que o espaço onde pretendem ir não estiver ocupado.

Ao longo deste trabalho será explicado o funcionamento da aplicação "Alarme Covid".

2. Sistema Desenvolvido

Para cumprir o objetivo descrito, a aplicação "Alarme Covid" permite que um utilizador realize as seguintes ações:

- Criar uma conta na aplicação, indicando o seu *username*, nome e a password a utilizar. Deverá indicar também se pretende subscrever o serviço que lhe dará benefícios na aplicação;
- Autenticar-se na aplicação;
- Informar a sua localização;
- Perceber a quantidade de pessoas numa dada localização;
- Receber uma notificação quando um determinado local estiver vazio;
- Receber notificações a avisar que esteve no mesmo local de uma pessoa doente;
- Comunicar que está doente;
- Receber um mapa com todos os locais já registados, bem como o número de pessoas doentes ou não que estiveram nesse local (Só para clientes que assinaram o serviço *plus*).

De modo a que os requisitos anteriores sejam alcançados, a aplicação apresenta a arquitetura descrita no diagrama de classes seguinte:

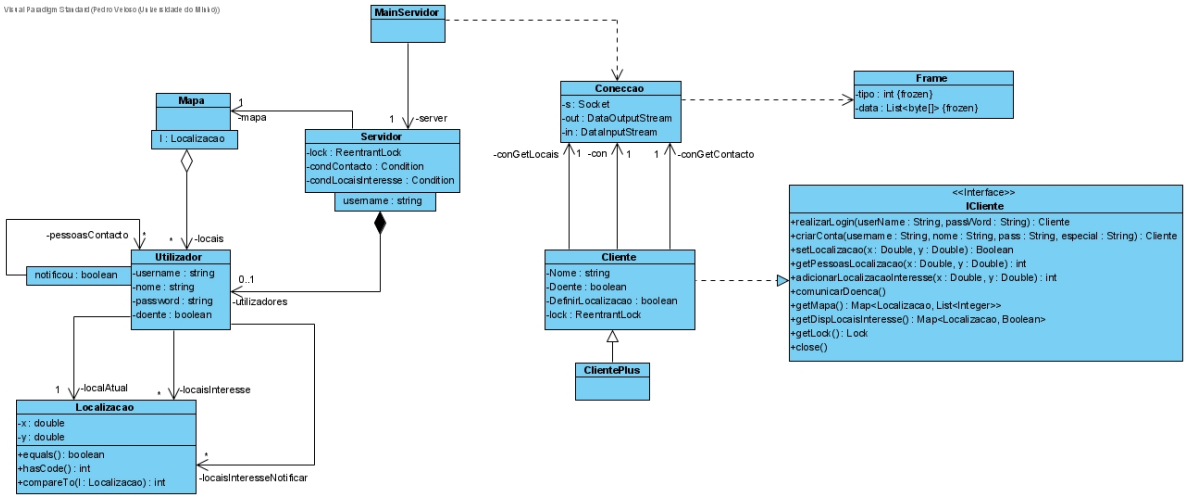

Figura 1: Diagrama de Classes

No servidor existe uma classe "Servidor" responsável por realizar todos os métodos necessários para os pedidos do cliente. É também na classe "Servidor" que estão copuladas todas as estruturas de dados (Mapa, Map<String,Utilizador> com todos os utilizadores registados na aplicação, …) que permitem realizar os métodos. Para suportar pedidos de vários clientes em simultâneo a classe "MainServidor" cria múltiplos threads cada um deles responsável por dar suporte a um cliente.

A classe "Cliente" é responsável por realizar os pedidos de cada método ao servidor. Contém ainda o *username* do cliente e a informação do seu estado de saúde. Como existe um tipo de cliente especial criou-se a classe "ClientePlus" que generaliza a classe "Cliente".

A classe "Coneccao" é responsável por receber um *frame* e escrevê-lo no *socket* (em formato binário) ou por receber dados binários e transformá-los em *frames*.

2.1. Cliente

A aplicação que permite o cliente comunicar com o servidor, foi desenvolvida adotando o padrão MVC, o que permite separar o código referente às *views*, *controllers* e à logica de negócio.

A lógica de negócio faz uso de uma classe que implementa a interface ICliente, que por sua vez, encarrega-se de fazer a ligação ao servidor através de *sockets* TCP, possuindo a codificação de todos os métodos referentes às ações que os utilizadores podem efetuar. Essa classe é também responsável por receber as notificações a comunicar ao utilizador.

Para cada ação pretendida por um cliente, deve ser criado um *frame* onde consta a informação necessária para o servidor a realizar. Este *frame* será enviado ao servidor, através das conexões existentes no cliente. Cada método no cliente origina um *frame* diferente, explicado em seguida.

Criar Conta

O método responsável por criar uma conta, gera um *frame* com o *username*, nome e a password a utilizar. Deverá conter ainda o resultado da subscrição do serviço que lhe dará

benefícios na aplicação. O *frame* é etiquetado com o valor 2, tal como demonstrado na Figura 2.

frame
Tipo = 2 Data = {nome, *username*, *password*, subscreveu}

Figura 2: Frame Criar Conta – Cliente

Realizar Login

O método responsável por realizar um login envia um frame ao servidor, com a seguinte estrutura:

frame
Tipo = 1 Data = {*username*, *password*}

Figura 3: Frame Realizar Login – Cliente

Definir Localização

O método responsável por definir a localização de um utilizador, cria um *frame* de tipo 3, juntamente com o *username* do utilizador e as novas coordenadas em X e Y.

Obter a quantidade de pessoas numa localização

Para obter a quantidade de pessoas presentes numa localização específica, é necessário enviar um *frame* ao servidor com a seguinte estrutura:

frame
Tipo = 4 Data = {*username*, *valor x*, valor y}

Figura 4: Frame Quantidade de Pessoas – Cliente

Obter mapa

Apenas um utilizador especial pode enviar uma *frame* do tipo 5 ao servidor para obter um mapa. Este *frame* não necessita de mais informação além da sua etiqueta.

Definir um local de interesse

Para adicionar um local de interesse, é necessário enviar o *frame* seguinte ao servidor:

frame
Tipo = 7 Data = {*username*, *valor x*, valor y}

Figura 5: Frame Definir Local – Cliente

Comunicar Doença

O utilizador informa o servidor que está doente, através do seguinte *frame*:

frame
Tipo = 6 Data = {*username*}

Figura 6: Frame Comunicar Doença – Cliente

Verificar disponibilidade dos locais de interesse

Para verificar a disponibilidade dos seus locais de interesse, o utilizador envia um frame ao servidor do tipo 8 com o seu *username*.

Receber Notificações

Ao criar a classe cliente, são criados 2 *threads* responsáveis por enviar 1 *frame* cada um. O *frame* pode ser do tipo 10 quando o objetivo for identificar um contacto próximo com uma pessoa infetada ou do tipo 11 para descobrir se o local de interesse está vazio. Estes *threads* ficam ativos, contudo adormecidos enquanto não existir informação a notificar.

frame
Tipo = 10 || 11 Data = {*username*}

Figura 7: Frame Receber Notificações - Cliente

2.2. Servidor

Do lado do servidor, a *main* fica à espera de uma nova conexão através de um *socket* TCP, criando uma *OverwatchThread* para cada cliente que efetue uma ligação.

Cada *thread* deverá ler a informação recebida, obtendo um *frame*. Com um *frame* para tratar, o *thread* deverá verificar a sua etiqueta e consoante o seu valor realizar uma ação.

Para cada ação, responde com diferentes *frames*, apresentados em seguida.

Criar Conta

Após realizar a ação de criar uma conta, o servidor responde ao cliente com um *frame* do tipo 2, que contém o resultado da criação da conta. Caso não seja possível criar conta, informa o motivo para esse insucesso.

frame
Tipo = 2 Data = {sucesso} || {sucesso, erro}

Figura 8: Frame Criar Conta - Servidor

Realizar Login

Ao realizar o login, o servidor informa o cliente sobre o resultado da tentativa. Se o login for efetuado com sucesso, indica ainda o tipo de subscrição e o estado do utilizador na aplicação (doente ou não). No caso do login de o login falhar, deve informar o motivo.

frame
Tipo = 1 Data = {sucesso, subscrição, doente} || {sucesso, erro}

Figura 9: Frame Realizar Login – Servidor

Definir Localização

Após a tentativa de definir a localização de um cliente, o servidor informa se esta foi bem-sucedida ou não, através de um *frame* do tipo 3.

Informar a quantidade de pessoas numa localização

Após receber um *frame* do tipo 4 ou 7 o servidor realiza o método correspondente a cada um dos tipos e envia ao cliente do mesmo tipo do recebido, bem como o número de pessoas presentes na localização recebida.

frame
Tipo = 4 || 7 Data = {número de pessoas}

Figura 10: Frame Quantidade Pessoas – Servidor

Disponibilizar mapa

Para disponibilizar o mapa, o servidor envia o seguinte *frame*:

frame
Tipo = 5 Data = {tamanho lista, [valor x, valor y, número de doentes, número de utilizadores]}

Figura 11: Frame Disponibilizar Mapa - Servidor

Notificar utilizador de um contacto

Sempre que um utilizador esteve em contacto com outro utilizador doente, o cliente recebe um *frame* do tipo 10 com a informação de realizou um contacto de risco.

Notificar utilizador da disponibilidade de uma localização de interesse

Para notificar o utilizador dos locais de interesse vazios, o servidor envia o seguinte *frame*:

frame
Tipo = 11 Data = {tamanho lista, [valor x, valor y]}

Figura 12: Frame Notificar Disponibilidade – Servidor

Informar disponibilidade dos locais de interesse

Para informar o utilizador da disponibilidade dos seus locais de interesse, o servidor envia o seguinte *frame*:

frame
Tipo = 8 Data = {tamanho lista, [valor x, valor y, vazia]}

Figura 13: Frame Informar Disponibilidade - Servidor

3. Conclusão

No contexto atual, com o número crescente de casos diários, é fundamental controlar o número de contágios. Para tal, a existência de uma aplicação de rastreio da localização das pessoas, para facilitar a identificação de contactos próximos, bem como evitar aglomerados de pessoas é fundamental.

Neste sentido, este trabalho tinha como objetivo o desenvolvimento de uma aplicação informática que auxilie nesse sentido. Para tal, foi identificado um conjunto de requisitos básicos e adicionais de modo a que o proveito retirado da aplicação a desenvolver fosse maximizado. Além disso, foi desenvolvida uma arquitetura que permite, através das estruturas de dados, guardar em memória toda a informação proveniente dos clientes. Para a informação chegar ao servidor utiliza-se *sockets* TCP, escrevendo a informação nos mesmos em formato binário. Após realizar ações o servidor volta a escrever no *socket* as respostas apropriadas.

Com todos os requisitos alcançados faz-se um balanço positivo do trabalho desenvolvido.