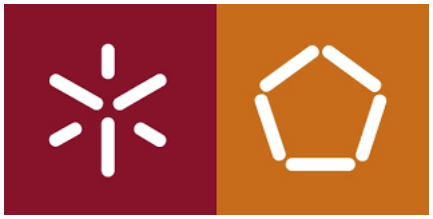

Universidade do Minho

Licenciatura em Engenharia Informática

Sistemas Distribuídos

Trabalho Prático - Gestão de Frota

Grupo 9

Ana Rita Poças (A97284) Inês Ferreira (A97372)
Joana Branco (A96584) João Pedro Braga (A97368)

9 de janeiro de 2023

Introdução

Este trabalho prático surge no âmbito da Unidade Curricular, em que nos foi proposto a implementação de uma plataforma de gestão de uma frota de trotinetes elétricas, sob a forma de um par cliente-servidor em Java através do uso de *sockets* e *threads*.

Os objetivos definidos para este projeto baseiam-se nas funcionalidades a implementar descritas no enunciado. Estas focam-se principalmente na interação entre o cliente e o servidor que tem como finalidade apresentar as trotinetes disponíveis para o destino pretendido.

Num estado inicial deve ser possível a autenticação e registo do utilizador, permitindo-lhe efetuar login ou criar conta. Devem também ser consideradas as trotinentes existentes no mapa e se estas continuam disponíveis ou não. Adicionalmente, é gerado um sistema de recompensas que tem por base avaliar a distribuição das trotinetes disponíveis, ou seja, é gerada uma recompensa quando uma trotinete é movimentada de um local para outro. Por outro lado, deve ser registada a reserva de uma trotinete disponível, retornando o código de reserva e o seu local. O custo da viagem também deve ser calculado de acordo com o tempo passado desde a reserva e a distância percorrida pela trotinete. Numa fase final, devem ser implementadas as notificações que destinam-se a informar o utilizador de recompensas com origem a menos de uma distância fixa.

No contexto do nosso programa, definimos uma grelha 20x20 para o mapa, e as localizações são representadas por um par de inteiros (x,y).

Funcionalidades Implementadas

A aplicação distribuída que concebemos foi capaz de implementar as seguintes funcionalidades:

Autenticação e registo de utilizador

Um utilizador, da nossa aplicação, consegue comunicar com o servidor. Um utilizador tem, associado a si um username e uma password. Caso seja a sua primeira vez no programa, o utilizador deve registar a conta nova introduzindo os dados que pretende que fiquem associados a si. Depois de validados, os dados do utilizador são armazenados (através da funcionalidade de serialização - que permite que caso haja falha no servidor, os dados fiquem armazenados). A conexão entre o utilizador e o servidor é assegurada pela classe **Desmultiplexer** e pela classe **Connection**.

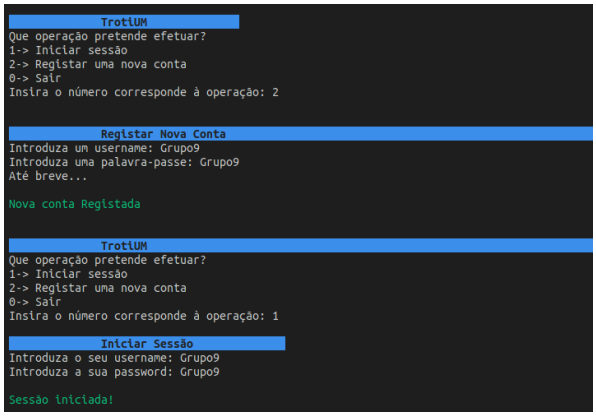

Figura 1: Menu de registo de conta e inicio de sessão

A classe **Connection** é uma intermediária entre os utilizadores e os servidores, uma vez que consegue enviar uma mensagem recebida através de um socket que lhe está associado ou receber dados por esse mesmo socket,transformando-os num formato que o utilizador e o servidor consigam ler, através do uso de DataInputStreams e DataOutputStreams.

A classe **Demultiplexer**, baseia-se na ligação entre o utilizador e a Connection. Uma vez que existem vários threads **Client**, que podem ter de esperar por uma mensagem, o **Demultiplexer**, garante que estes threads "adormeçam"enquanto esperam pela mensagem e sejam "acordados"assim que ela chegue. A existência de um desmultiplexador no nosso projeto faz com que consigamos receber dados de um canal e distribuir por vários canais. Para cada tag é criado um buffer de mensagens .

Listagem dos locais onde existem trotinetes livres

Depois de ter iniciado sessão, o programa pede ao utilizador que introduza a sua localização atual. Esta localização é enviada para o servidor, que se encarrega de verificar as trotinetes livres num raio de D=2. Para tal, foi necessário implementarmos as classes **Positions** e **Trotinetes**.

A classe **Positions** encarrega-se de guardar a informação relativa às coordenadas existentes no nosso programa. A classe **Trotinetes**, inicializa as trotinetes existentes no nosso mapa e implementa todos os métodos relativos à manipulação das trotinetes na nossa aplicação. Assim, de forma a conseguirmos verificar as trotinetes que estão num raio de D=2 da posição do cliente, recorremos à função *getClosestTrotinetes* que utiliza a distância de Manhattan para efetuar o cálculo. Essa informação é depois enviada para o cliente, sendo imprimida no ecrã.

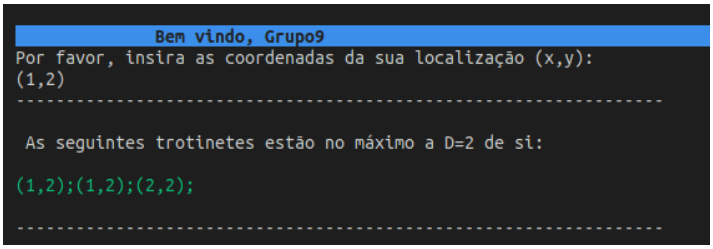

Figura 2: Menu de listagem das trotinetes livres num raio de D=2

Listagem das Recompensas

Depois da apresentação no ecrã do locais das trotinetes que se encontram livres, procedemos, também, à apresentação no ecrã da listagens das origem-destino que dão direito às recompensas. O servidor, caso existam recompensas, envia as listas de origens (posições que possuem mais do que uma trotinete) e destinos (posições sem nenhuma trotinete num raio D=2) elegíveis para recompensas.

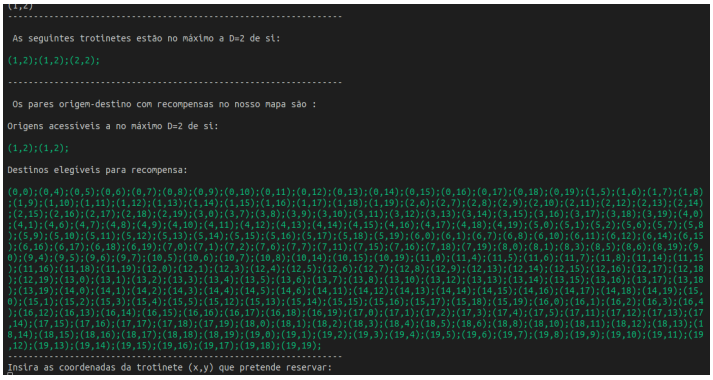

Figura 3: Menu de Listagem das Recompensas

Reserva de trotinetes livres

Depois de ter recebido a listagem das trotinetes livres num raio D=2, o programa pede ao cliente que escolha a que pretende reservar. O cliente deve selecionar a trotinete que pretende e esta informação será enviada para o servidor. O servidor, irá guardar este valor como a origem do percurso do cliente e terá a atenção de remover esta trotinete da lista de trotinetes livres, uma vez que está agora a ser usada por este cliente.

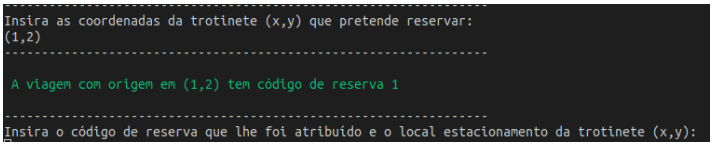

Figura 4: Menu de reserva de trotinete livre

Estacionamento de uma trotinete

Para estacionar uma trotinete, o utilizador deve fornecer o código de reserva que lhe foi atribuído anteriormente e a posição onde pretende estacionar a sua trotinete. O cliente fornece esta informação ao servidor, que calculará o custo da viagem, com base na distância percorrida entre a origem e o destino (local de estacionamento da trotinete) e o tempo percorrido desde a reserva. O servidor, calculará também a recompensa, caso a posição onde o utilizador pretende estacionar a trotinete seja apta para recompensa. Desta forma é apresentado ao utilizador o valor do custo da viagem e a o valor da recompensa (0, caso não exista).

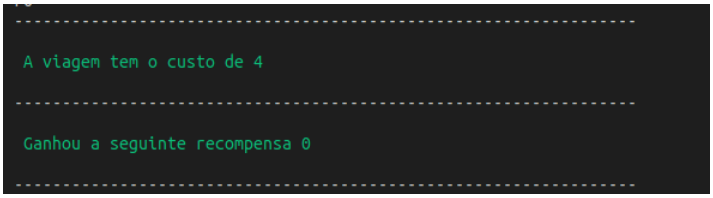

Figura 5: Menu de estacionamento de uma trotinete

Observações

Mecanismos de Exclusão Mútua no nosso programa

Sendo o trabalho desenvolvido uma aplicação distribuída, um dos principais objetivos é que suporte a programação concorrente. Para tal, foram adotados os seguintes mecanismos para garantir quenão temos problemas de exclusão mútua ocorre no nosso programa:

- Tivemos que ter atenção a fazer um locks na trotinete, para as conseguir reservar;
- Usamos locks nas classes Users, Trotineres, Recompensa, CodUser e nas estruturas de dados do Servidor;
- Para garantirmos exlusão mútua nas contas usamos ReadWriteLocks pois fazemos várias leitruas e poucas escritas e assim melhoramos o desempesenho do nosso programa;
- Utilizamos condition() e await() e signal(), , na classe das Recompensas;

Especificação das Frames

A comunicação entre o utilizador e o servidor, recorremos à representação das mensagens trocadas entre eles por frames. Uma frame é constituída por uma *tag*- valor numérico, identificador do cliente - usada pelo servidor para saber qual o cliente que comunicou com ele e uma mensagem- array de bytes. Na tabela abaixo temos a representação de cada valor de *tag*:

Tag	**Tipo de mensagem**
0	Solicitação de inicio de sessão
1	Solicitação de novo resgisto de conta
2	Inserir coordenadas do utilizador no mapa
3	Pedido de reserva de trotinete
4	Inserir o código de reserva coordenadas do destino do utilizador no mapa
5	Listar as recompensas presentes no mapa

Conclusão

Por fim, acreditamos que fomos capazes de desenvolver as funcionalidades básicas propostas e consolidar os conhecimentos obtidos no desenvolver da Unidade Curricular. O projeto também permitiu também aprimorar as nossas noções relativos a threads, sockets, etc.

Apesar de não termos sido capazes de implementar a funcionalidade de "Um cliente pedir para ser notificado quando apareçam recompensas com origem a menos de uma distância fixa D de determinado local", uma vez que todas as outras funcionalidades estão completamente implementadas, acreditamos que o programa está totalmente de acordo com as nossas expectativas, pois achamos que fizemos um bom desempenho e conseguimos apresentar a implementação da autenticação e registo de um utilizador, listar os locais onde há trotinetes disponíveis e as respetivas recompensas.