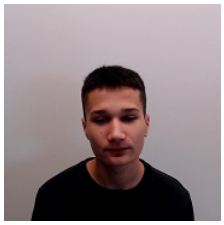
Universidade do Minho
Escola de Engenharia

Sistemas Distribuídos

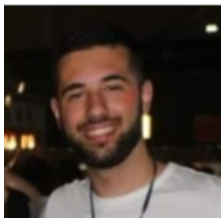
Benjamim Meleiro Rodrigues A93323
Francisco Pinto Lameirão A97504
Lara Beatriz Pinto Ferreira A95454
Luís Miguel Moreira Ferreira A95111



A93323

A97504

A95454



A95111

9 de janeiro de 2022

Índice

1 Introdução

Neste trabalho, realizado no âmbito da disciplina de Sistemas Distribuídos, foi-nos proposto desenvolver e implementar uma plataforma de gestão de uma frota de trotinetes elétricas, sob a forma de um par cliente-servidor em Java utilizando sockets e threads.

O objetivo deste trabalho é permitir aos utilizadores reservar e estacionar trotinetes em diferentes locais, tendo ou não a possibilidade de serem recompensados dependendo do local onde estacionam as mesmas.

2 Classes Implementadas

2.1 Servidor

- **Server**
  Esta classe é responsável por estabelecer uma conexão com cada client através da criação de uma thread para cada um. Ela também inicia a thread associada a um objeto da classe RewardManager.

- **ServerManager**
  Esta classe armazena todas as estruturas de dados e métodos necessários para gerenciar e atender às funcionalidades do servidor. Ela é responsável por prover todas as operações e recursos necessários para o servidor funcionar corretamente. De forma a termos estruturas de dados mais completas existem duas Inner classes nesta Classe: a classe Trotinete e a classe User. As Collections usadas são também extendidas em classes de forma a implementarmos um ReentrantLock para cada classe.

- **NotificationManager**
  Esta classe é responsável por enviar as notificações para o cliente quando este escolhe "subscrever-se"neste serviço. Quando o cliente faz a subscrição é iniciada uma Thread associada a um Objeto desta classe.

- **RewardManager**
  Esta classe é responsável por gerenciar o mapa de recompensas.

2.2 Cliente

- **Client**
  Esta classe é responsável por estabelecer a conexão entre o cliente e o servidor. Ela inicia duas threads, uma associada a um objeto da classe Reader e outra associada a um objeto da classe Writer. A classe Reader é responsável por ler as mensagens enviadas pelo cliente, enquanto a classe Writer é responsável por enviar mensagens para o cliente.

- **ReadData**
  Esta classe é responsável por ler as respostas enviadas pelo servidor após o utilizador realizar o login, e exibir essas respostas para o utilizador através do Stdout (saída padrão).

- **WriteData**
  Esta classe é responsável por enviar e ler dados do servidor até o utilizador realizar o login. Após o login, a classe continua a ser responsável por enviar mensagens para o servidor, mas a função de ler as respostas passa a ser de responsabilidade do ReadData.

2.3 Classes auxiliares

- **Point**
  Classe usada para representar um ponto num local no espaço de coordenadas (x,y).

- **Reward**
  Classe usada para representar um Reward, constituido por dois Points. Um dos Points representa a origem do Reward e o outro o destino.

3 Funcionalidades do Programa

3.1 Login

Para efetuar login, o utilizador irá introduzir o *username* e a *password*. Estas serão passadas ao servidor, que irá analisar a mensagem e fazer o login do Utilizador casos os dados de login estejam corretos. Se algum dos dados estiver incorreto, a mensagem de erro correspondente será transmitida, atravás do uso de Exceptions.

3.2 Registo

Para efetuar registo, o utilizador irá introduzir o *username* e a *password*. A conta será adicionada ao UsersMap no ServerManager de forma a ser guardada.

3.3 Procurar Trotinete

Esta operação é chamada caso seja escolhida a opção 1 no menu depois de fazer login.

O servidor chama o método listFreeTrotinetesOnCoords. Este método cria um array com todas as trotinetes livres que estejam a uma distância fixa das coordenadas dadas pelo Cliente.

O servidor manda então uma mensagem para o Cliente, avisando que não existem trotinetes por perto ou apresentando todas as trotinetes próximas em forma de string dependendo se há ou não trotinetes livres na zona.

3.4 Procurar Recompensas

Esta operação é chamada caso seja escolhida a opção 2 no menu depois de fazer login.

O servidor chama o método listRewardsOnCoords. Este método cria um array com todas as recompensas que estejam a uma distância fixa das coordenadas dadas pelo Cliente.

Depois, caso o array esteja vazio, o Server envia para o Cliente o aviso que não existem recompensas por perto, caso tenha algum elemento, o servidor transforma as recompensas em strings e envia-as em forma de String para o Cliente.

3.5 Reservar Trotinete

Esta operação é chamada caso seja escolhida a opção 3 no menu depois de fazer login.

O servidor chama o método reservaTrotinete. Este método procura a trotinete mais próxima das coordenadas dadas pelo Cliente que esteja livre e que não exceda uma distância máxima fixa.

Se existir uma trotinete que satisfaça estas condições, o Servidor envia para o Cliente a mensagem a avisar que a trotinete foi reservada e o código associado a esta reserva. Se não existirem, o servidor simplesmente avisa que não há trotinetes próximas que estejam disponíveis.

3.6 Estacionar Trotinete

Esta operação é chamada caso seja escolhida a opção 4 no menu depois de fazer login.

O servidor chama o método estacionaTrotinete. Este método verifica se o estacionamento da trotinete é válido e, caso seja, calcula o custo ou recompensa da viagem, dependendo se esta viagem se encontra ou não nas Recompensas.

O Servidor envia então uma mensagem para o Cliente, informando-o do custo/recompensa da viagem ou da inexistência da Reserva da Trotinete que o Cliente queria estacionar.

3.7 Ser notificado de Recompensas

Esta operação é chamada caso seja escolhida a opção 5 no menu depois de fazer login.

O servidor inicia uma thread apartir dum objeto da classe NotificationManager, que irá enviar para o Cliente notificações de recompensas sempre que forem encontradas Recompensas nas coordenadas dadas por este.

O Servidor manda uma mensagem para o Cliente informando se esta operação foi realizada ou não.

3.8 Deixar de ser notificado de Recompensas

Esta operação é chamada caso seja escolhida a opção 6 no menu depois de fazer login.

Caso a thread do NotificationManager esteja a correr o Servidor dá interrupt a esta.

O Servidor manda uma mensagem para o Cliente informando se esta operação foi realizada ou não.

3.9 Exit

O Cliente e o Server fecham as sockets correspondentes e são terminadas as threads associadas caso estejam a ser corridas, tal como a thread do Servidor NotificationManager e as threads ReadData e WriteData do Cliente.

4 Conclusão

Com a realização deste trabalho prático foi-nos permitido consolidar a matéria que estava a ser lecionada nas aulas de Sistemas Distribuídos, nomeadamente sobre Sockets e Threads, e a comunicação entre cliente-servidor, recorrendo a Sockets TCP.

No geral, achamos que, através deste trabalho, conseguimos aplicar os conceitos acima descritos num contexto de uma aplicação e, assim, ficamos a entender muito melhor como usar threads num programa e como criar as comunicações entre um cliente e um servidor de qualquer aplicação.