Universidade do Minho
Departamento de Informática

Trabalho Prático
Sistemas Distribuídos

Plataforma de Gestão de Frota de Trotinetes Elétricas

Repositório do GitHub:
https://github.com/goncalobraga27/SD.git

Grupo 37:
Gonçalo Braga (A97541)
João Gonçalves (A95019)
João Martins (A96215)

9 de janeiro de 2023

1 Introdução

No âmbito da cadeira de Sistemas Distribuídos, foi-nos pedido para implementarmos uma plataforma de gestão de uma frota de trotinetes elétricas. O requisito principal é esta ser desenvolvida sob a forma de um par cliente-servidor em Java, recorrendo a *sockets* TCP e *threads*, utilizando primitivas de controlo de concorrência e exclusão mutua como locks e variáveis de condição.

2 Funcionamento do Serviço

O nosso serviço funciona, muito resumidamente, com base em mensagens trocadas entre Cliente e Servidor, utilizando *tags* em cada mensagem enviada por um Utilizador para o Servidor, para assim o Servidor conseguir realizar a funcionalidade pedida. Assumimos que para as funcionalidades respetivas, o D tem um valor de 2, o N tem um valor de 20, e o S tem um valor de 100.

MÉTODO	TAG	PEDIDO	RESPOSTA
login	1	User	int
register	2	User	int
freeScotters	3	Point	PointList
getRewards	4	Point	RewardList
newRservation	5	Point	Code,Local
concludeReservation	6	Point,Code	Boolean,Double
getNotifications	7	User,Point	RewardList
cancelNotifications	8	User	——-

3 Arquitetura da Aplicação

A arquitetura da aplicação é a seguinte:

- Cliente - O cliente é um cliente multithreaded, isto é, pode fazer diferentes threads podem utilizar a mesma ligação para fazer pedidos ao servidor. Além disso o cliente faz uso de um desmultiplexador.
- Servidor - O servidor foi concebido da seguinte maneira: uma thread por pedido. Sendo assim, sempre que um pedido com uma determinada tag chega ao servidor, este envia esse pedido para a thread responsável pela satisfação dos pedidos com a tag recebida.

3.1 Funcionalidades

Em termos de funcionalidades, um utilizador da nossa plataforma consegue:

3.1.1 Registar e fazer login na plataforma

Quando um utilizador entra na aplicação, pode escolher fazer login ou registar-se.

Se o utilizador decidir registar um novo user na aplicação, terá de colocar o novo username e password, sendo essa mensagem enviada para o servidor. De forma seguida o cliente recebe uma resposta, 0 ou 1, indicando sucesso ou insucesso no pedido.

Para fazer login na aplicação, o utilizador deverá indicar o username e a password associados à conta. O Servidor irá verificar a existência do username e password e se já não existe uma conexão ativa desse utilizador, e se essa verificação for verdadeira, irá enviar para o cliente uma resposta, 0 ou 1, indicando sucesso ou insucesso no pedido.

Desta forma, conseguimos perceber como é feito o login e o registo de um jogador na plataforma.

3.1.2 Pedir para listar os locais onde existem trotinetes livres

Quando no menu principal da aplicação, o utilizador pode escolher listar os locais onde existem trotinetes disponiveís para serem reservadas. Sendo assim o cliente, envia uma mensagem para o servidor com as coordenadas (x,y) de onde se encontra.

O servidor com a mensagem recebida do cliente, que é composta por: tag e coordenadas, consegue satisfazer este pedido do cliente, uma vez que sabe qual o número da tag enviada.

Com as coordenadas, o servidor lista o local das trotinetes, na qual usamos uma *PointList* (lista de locais com métodos específicos de serialização), que se encontram a menos de uma distância fixa D das coordenadas inseridas pelo cliente.

Depois desta listagem do local das trotinetes pelo servidor, este envia esses dados para o cliente, com a mesma tag.

3.1.3 Pedir para listar as recompensas

Quando no menu principal da aplicação, o utilizador pode escolher listar os locais onde existem recompensas dado uma certa coordenada (x,y).

Para isso, o cliente envia uma mensagem para o Servidor com as coordenadas que pretende para consultar as recompensas dísponiveis.

O Servidor, recebendo esta mensagem irá tratar de obter as recompensas que se encontram a menos de uma distância fixa D das coordenadas inseridas pelo utilizador. Depois disso, envia a *RewardList* como resposta para o Cliente

3.1.4 Reservar trotinete

Para ser reservada uma trotinete no sistema,o cliente apenas tem de inserir as coordenadas do local onde se encontra.

Com estas coordenadas, o servidor procura a trotinete livre mais próxima limitada a uma distância fixa D. Caso encontre uma trotinete, o servidor cria uma nova reserva e envia para o servidor o código da reserva e o local da trotinete reservada. Caso contrário, envia no valor do código o valor 0, visto que, os códigos das reservas são sempre superiores a 1, premitindo ao cliente fazer a desserialização corretamente. Também pode indicar código de erro -1, se o ponto introduzido não se encontrar nos limites do mapa.

Com esta explicação, conseguimos perceber qual é o fluxo de dados que existe nesta opção da nossa aplicação.

3.1.5 Estacionar trotinete

Se o cliente desejar estacionar uma trotinete reservada, terá de indicar o código da reserva já efetuada e o local para onde quer estacionar a trotinete. Para isso, envia uma mensagem serializada para o Servidor com esses valores, e quando chega ao Servidor este tem de desserializá-la, lendo as coordenadas e o código de reserva inserido pelo utilizador.

De seguida, é enviado um tuplo onde o primeiro elemento corresponde a um valor booleano que indica se a viagem se trata de uma recompensa ou não, e o segundo elemento varia com o valor da variável booleana, sendo que se for true corresponde ao valor da recompensa, enquanto que se for false corresponde ao valor da viagem até esse local. Para concluir, o servidor envia para o cliente o código de reserva associado à viagem, e de seguida envia o *boolean* e o valor. Caso contrário, envia o código de reserva igual a 0, indicando que não existe uma reserva com aquele código, ou igual a -1, indicando que o ponto introduzido não se encontra nos limites do mapa.

3.1.6 Pedir para ser notificado quando aparece uma determinada Recompensa

Como a gestão de recompensas acontece sempre em background então iremos ter a emissão de recompensas por parte do sistema, enquanto que são atendidos diferentes tipos de pedidos.

Sendo assim , para receber notificações terá de indicar o local na qual quando apareçam recompensas com origem a menos de uma distância fixa D desse local seja notificado. No nosso cliente uma thread é iniciada à espera que receba as notificações. Na parte do servidor quando este pedido é feito as notificações são ativas e o servidor entra num ciclo enquanto aquele utilizador tiver as notificações ativas. A cada interação é enviada uma lista de recompensas geradas, sendo que o servidor fica à espera enquanto o tamanho da lista de recompensas não se altera.

3.1.7 Cancelar pedidos de notificação

De forma contrária, para o cancelamento das recompensas,o utilizador apenas expressa o seu pedido, e o sistema irá inferir os dados de sessão do utilizador, e irá enviá-los para o servidor.

O servidor ao receber esses dados, com a tag número 8, apenas irá desativar as notificações desse utilizador.

3.2 Geração de Recompensas

As recompensas no programa foram pensadas para permitirem uma melhor distribuição das trotinetes pelo "mapa".

Sendo assim, as recompensas são geradas em duas fases:

- Nova Reserva - Existe uma nova trotinete ocupada e, assim sendo, verificamos se existe uma trotinete livre a uma distância fica D dessa trotinete. Caso não exista vamos gerar recompensas de todas as trotinetes livres para a posição desta trotinete ocupada.

- Estacionar Trotinete - Existe uma nova trotinete livre e, assim sendo, verificamos para cada posição do mapa se exista alguma trotinete livre a uma distância inferior ou igual a D. Caso não exista geramos uma recompensa a partir da posição da trotinete livre para a posição do mapa.

4 Exclusão Mutua e Controlo de Concorrência

No servidor possuímos o estado partilhado *Management* na qual existem as seguintes estruturas:

- Scooter[] - Possuímos um lock por Scooter sem necessidade de um lock para a estrutura pois o número se trotinetes mantêm-se;
- List <Reward> - Possuímos um lock para a estrutura, visto que as operações são de inserção remoção e de consulta e o objeto Reward é um objeto imutável;
- Map<Integer,Reservation> - Possuímos um lock para a estrutura, visto que as opereções são de inserção e remoção.
- Map<String, User> - Possuímos um ReadWrite lock, visto que a grande maioria das operações são de leitura, em particular, de autenticação dos utilizadores.

Para além disto usamos duas variáveis de condição, uma associada ao lock de recompensas, necessária para notificar clientes à espera de notificações e outra associada a outro lock usado para gerar recompensas.

4.1 Scooter

Esta classe, como o próprio nome indica, representa as trotinetes no sistema. Sendo assim, cada trotinete possui a posição onde se encontra, o estado de ocupação e o Lock respetivo.
Em muitos métodos acima referidos é necessario adquirir o lock da coleção de trotinetes, logo são adquiridos sempre pela mesma ordem, ou seja, ordem do *array* e libertados um a um o mais cedo possível, recorrendo à estratégia do *two phase locking*.

4.2 Reservation

Esta classe representa uma reserva de uma determinada trotinete. Sendo assim a reservation possui o código da reserva, a data de reserva, a trotinete que foi reservada. Também possuímos um lock estático para obter e encrementar o código da próxima reserva.

4.3 Reward

Esta classe representa uma recompensa e possui o ponto de inicial e o ponto final, que são variáveis imutáveis. Nesta classe não registamos o valor da recompensa, visto que só é calculado no momento de estacionar uma trotinete.

4.4 Utilizador

Esta classe representa um utilizador do programa. Sendo assim um utilizador é identificado por um nome de utilizador e por uma password. Além desta caracterização do cliente, esta classe possui um boolean que serve para identificar se o utilizador quer receber ou não notificações do programa.

4.5 Posições

No estado partilhado guardamos duas posições com um lock associado. Essas posições são a posição da última trotinete ocupada e a posição da última trotinete estacionada. Assim é possível gerar recompensas com estas duas posições como referido anteriormente.

4.6 Biblioteca do Cliente

A Biblioteca do Cliente é representada como uma ponte entre a interface do Utilizador e o Servidor, sendo que os métodos que lá se encontram servem para fazer os pedidos que o Cliente deseja de uma forma abstracta, sem o Cliente saber que está a interagir diretamente com o Servidor.

5 Implementação da Interface do Utilizador

Quanto à interface do Utilizador, é feita com base no terminal, imprimindo linhas de texto, onde o Utilizador consegue ter uma interação com a aplicação, com base na utilização de um Scanner.

6 Aspetos finais

Em suma, neste projeto foi-nos pedido a implementação de um sistema cliente-servidor, para uma plataforma de gestão de trotinetes elétricas, utilizando primitivas de exclusão mutua e controlo de concorrência, bem

como comunicação utilizando sockets TCP e serialização e desserialização dos dados enviados entre os dois processos.

Na nossa opinião, realizámos um bom trabalho em todos os aspetos mencionados em cima.Os aspetos que podem ser melhorados são a otimização na geração de recompensas e a limitação do número de reservas por cliente.