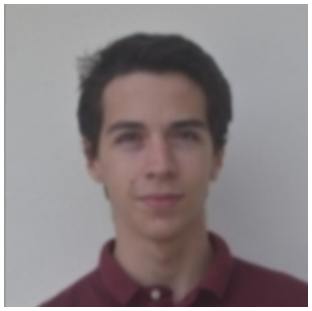
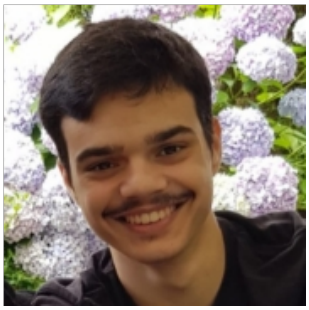
Gestão de Frota

Trabalho realizado por:

Bianca Araújo do Vale
Ricardo Lopes de Lucena
Ricardo Silva Machado Araújo
Xavier Santos Mota



A97746



A96394

A95835

A88220

Grupo 29
2022/2023
Sistemas Distribuídos
Universidade do Minho

Índice

Índice de Imagens

1 Introdução

O presente relatório destina-se à exposição e apresentação do trabalho prático proposto pelos docentes da Unidade Curricular de Sistemas Distribuídos.

De uma forma geral, foi-nos proposta a implementação de uma plataforma de **Gestão de uma Frota de Trotinetes Elétricas**. Este sistema irá funcionar sob a forma de um par cliente-servidor em Java utilizando *sockets* e *threads*.

A plataforma deve suportar diversas funções, nomeadamente, a autenticação e registo do utilizador, listagem dos locais onde existem trotinetes livres, listagem das recompensas, reserva de uma trotinete livre, estacionamento de uma trotinete e, por fim, a notificação ao cliente quando aparecerem recompensas.

2 Classes Implementadas

Neste tópico vamos fazer uma breve descrição das classes que decidimos implementar.

2.1 Cliente

A classe **Cliente** é responsável pelo programa principal da aplicação desenvolvida, que é o que os utilizadores usam para terem acesso às funcionalidades da mesma.

2.2 ClienteHandler

Esta classe é usada para criar uma ligação a um determinado servidor, através de um *socket*. Esta classe utiliza *threads* para realizar diversas operações, dependendo da opção escolhida pelo utilizador. Assim que o utilizador escolha a sair, fecha-se a ligação e o programa é encerrado.

2.3 Demultiplexer

Classe que é utilizada para controlar a comunicação entre dois sistemas e capaz de suportar clientes *multithreaded*, através da *TaggedConnection*. Esta classe tem como objetivo multiplexar e desmultiplexar *TaggedFrames* através da conexão *TaggedConnection*.

A classe *TaggedFrame* é uma estrutura de dados auxiliar, que é utilizada pela classe *Demultiplexer* para armazenar os *TaggedFrames* recebidos. Esta é composta por uma fila de *TaggedFrames* e uma *Condition*, que permite que as *threads* aguardem até que uma dada condição seja recebida.

2.4 Login

Esta classe é responsável por gerir o *login* e o registo dos utilizadores, armazenando os *usernames* e as respetivas *passwords* das contas num *Map*. Aqui é verificado se a *password* fornecida corresponde ao utilizador que tenta fazer *login*.

2.5 Mapa

Esta é uma classe responsável por gerar um mapa e guardar as posições das trotinetes.

2.6 Menu

Esta classe contém os diversos menus e os métodos para recolher a informação acerca dos mesmos.

2.7 Recompensa

A classe **Recompensa** representa uma recompensa que um determinado utilizador pode ganhar por transportar uma trotinete desde a origem até à localização objetivo da recompensa. Esta armazena informaçóes sobre a origem e o destino, assim como a distância entre estas duas localizações e até mesmo o valor da recompensa. O valor da recompensa é calculado em função da distância.

2.8 Server

Esta classe implementa um servidor. Quando se estabelece uma conexão, a classe cria uma nova *thread*, que é responsável por ler mensagens do cliente, descodificá-las e executar as respetivas ações, que incluem reservar ou estacionar uma dada trotinete, obter as localizações sobre trotinetes disponíveis e informações sobre as recompensas.

2.9 SistemaRecompensas

Esta classe gera um sistema de recompensas para os percursos realizados pelas trotinetes.

2.10 TaggedConnection

Esta classe é responsável por representar uma relação com um determinado servidor e envia e recebe dados através dessa relação.

2.11 Tuple

A classe **Tuple** representa um par ordenado de inteiros. Possui dois campos, x e y, que armazenam os valores do par ordenado.

3 Operações

3.1 Registar utilizador

Para realizar esta operação, o cliente deverá selecionar, no menu, a tecla 1. Aqui, vai ser verificadp se o cliente já está ou não registado. Caso não se encontre registado, procede ao registo da nova conta.

3.2 Login

Para efetuar o *login*, o cliente deverá selecionar a tecla 2 no menu e de seguida escrever o *username* e a respetiva *password*. É também importante afirmar que o cliente só tem acesso às outras funcionalidades do sistema se tiver efetuado antes o *login*. Esta operação só é efetuada caso o cliente já estiver resgistado.

3.3 Sair

Esta operação corresponde a selecionar a tecla 0 no menu e, de seguida, o programa encerra.

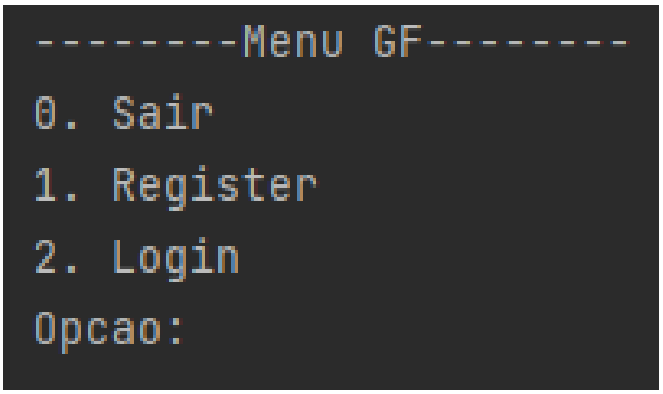


Figure 1: Menu para efetuar *login*

3.4 Listar os locais onde existem trotinetes livres

Após efetuado o *login*, o sistema apresenta uma lista das trotinetes disponíveis, mais perto, dentro de um raio N, cujo valor escolhido foi 2. É importante realçar que o sistema só apresenta esta lista após o cliente ter fornecido a sua localização atual.

3.5 Listar as recompensas

Posteriormente a realizar o *login*, o cliente fornece a sua localização e o sistema irá apresentar uma lista de locais que têm recompensa, caso o mesmo leve a trotinete para um desses locais. O sistema só apresenta os locais que estejam dentro de um raio N, cujo valor é 2.

3.6 Reservar uma trotinete livre

Com o objetivo de efetuar uma reserva de uma trotinete livre, depois de o *login* ter sido efetuado, o cliente indica ao servidor a trotinete que tenciona reservar. Caso não haja trotinetes disponíveis, o sistema apresenta um código de erro. Caso contrário, exibe a localização da respetiva trotinete e o código de sucesso.

3.7 Estacionar uma trotinete

Seguidamente a fazer *login*, com o intuito de estacionar uma trotinete, o cliente digita as coordenadas do local onde deseja estacionar a trotinete. Caso previamente o cliente tenha entrado numa recompensa, o sistema verifica se o local de estacionamento é o local destino da recompensa. Caso seja verídico, este apresenta o valor do custo da viagem já com a recompensa incluída.

3.8 Notificar cliente

Caso o cliente queira receber notificações sobre recompensas, liga, caso contrário, desliga. è de frizar que tal só acontece caso tenha sido efetuado o *login* previamente.

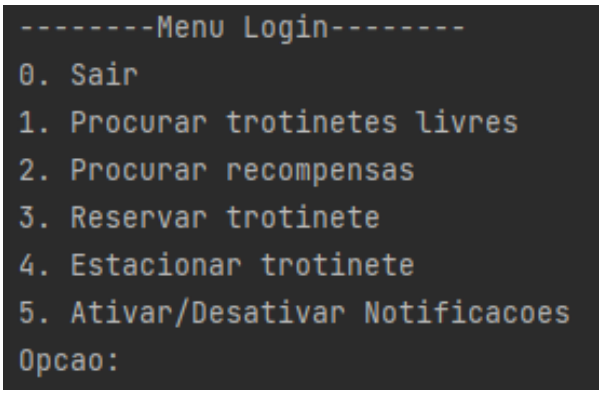


Figure 2: Menu Principal

4 Conclusão

Em modo de conclusão, a realização do trabalho prático da Unidade Curricular de Sistemas Distribuídos permitiu ao grupo a consolidação dos conhecimentos adquiridos nas aulas teóricas, bem como nas teórico-práticas.

O grupo sentiu algumas dificuldades aquando da realização do trabalho, como por exemplo, um erro que tínhamos no Demultiplexer e demoramos algum tempo para o encontrar e também sentimos alguma complexidade ao implementar o cliente *multithreaded*.

Assim, finalizado o trabalho e apesar das diversas dificuldades, o grupo está satisfeito com o resultado final, considerando, deste modo, ir de encontro ao que foi solicitado pelos docentes da respetiva Unidade Curricular.