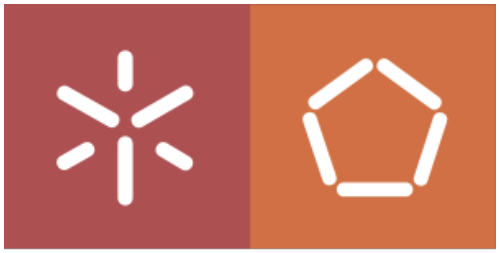


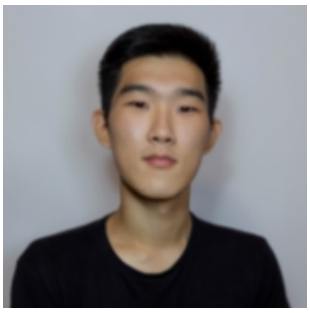
Universidade do Minho
Escola de Engenharia

Sistemas Distribuídos

Plataforma de Gestão de Trotinetes Elétricas

Gonçalo Santos A95354

Daniel Furtado A97327



Daniel Du A97763

Nuno Costa A96897

Grupo 33
9 de janeiro de 2023

Conteúdo

1 Introdução

Este relatório é relativo à elaboração do trabalho prático proposto na Unidade Curricular de *Sistemas Distribuídos* com o tema de *Gestão de Frotas de Trotinetes Elétricas*.

Este projeto, seguindo o enunciado do trabalho prático, terá de ser implementado na linguagem de programação *Java*, seguir a arquitetura cliente-servidor utilizando *sockets* e *threads* e com o objetivo principal permitir que os usuários reservem e estacionem as trotinetes em diferentes locais de um mapa.

O mapa da nossa plataforma é dividido numa grelha de N x N locais, sendo as coordenadas discretas e pares, a distância entre dois locais é calculada utilizando a *Distância Manhattan*.

Para além disso, existe um sistema de recompensas de forma a incentivar a boa distribuição das trotinetes pelo mapa em que as recompensas são identificadas por pares de posição origem e posição destino. O valor da recompensa é calculado em função da distância percorrida.

Deste modo, iremos nas secções seguintes abordar como o nosso grupo definiu a plataforma.

2 Descrição classes implementadas

2.1 Relacionadas com o Servidor

- **MainServidor -** Classe em que se efetua a ligação do Servidor ao Cliente e ocorre a inicialização da execução de uma thread associada a um objeto da classe *Sessão*.
- **Sessão -** A classe Sessão é a classe que representa a ligação entre o Servidor e um Cliente. Esta classe tem acesso à classe Servidor e um lock relacionado com a sessão. Para além disto, a sessão é responsável por atribuir o trabalho aos diferentes ServerWorkers.
- **ServerWorkers -** Cada um dos ServerWorkers está relacionado com uma funcionalidade da plataforma e têm como objetivo executa-la e enviar a resposta ao cliente. Estes têm acesso ao Servidor.
- **Servidor -** Classe responsável pela a gestão de todas as estruturas de dados partilhados, como por exemplo, os clientes ligados ao servidor, o mapa com as trotinetes, as diferentes recompensas disponíveis no momento e as reservas.
- **Mapa -** Classe responsável por guardar os dados do mapa e capaz de apresentar as trotinetes livres.
- **GereReserva -** Classe responável pela gestão das reservas que vão existindo ao longo do decorrer do programa;

- **GereRecompensa -** Classe encarregue de gerar as recompensas bem como a gestão de cada uma.
- **GereClientes -** Classe responsável pela gestão de Autenticação e Registo na plataforma.
- **GereNotificações -** Classe responsável pela gestão das notificações.

2.2 Relacionadas com o Cliente

- **MainCliente -** Classe responsável por enviar pedidos ao servidor e receber as respetivas respostas com auxílio da classe MenuCliente.
- **MenuCliente -** Classe responsável por escrever para o ecrã e receber os *inputs* do utilizador.
- **Demultiplexer -** Classe responsável por distribuir as respostas do servidor para as threads corretas do Cliente. Esta classe é obrigatória ter um Cliente multi-threaded.
- **ClientWorker -** Cada um dos ClientWorker está relacionado com uma funcionalidade da plataforma e têm como objetivo enviar pedidos e receber a resposta do servidor.

2.3 Relacionados com o Servidor e Cliente

- **Reserva -** Classe responsável por armazenar dados relativos a uma Reserva, como por exemplo: Trotinete associada, Código de Reserva, e Posições inicial e final.
- **Recompensa -** Classe responsável por armazenar dados relativos a uma Recompensa,como por exemplo: Posição Inicial, Posição final e valor;
- **Estacionamento -** Classe responsável por armazenar dados de um Estacionamento, tais como: Código de Reserva e Posição de Estacionamento;
- **Posição -** Classe que representa uma posição no mapa;
- **Trotinete -** Classe responsável por armazenar dados relativos a uma trotinete, como por exemplo: um id, a sua posição e um boolean que representa se está livre ou não;
- **Cliente -** Classe que permite armazenar dados de um Cliente, tal como: Nome de Utilizador e Password
- **TaggedConnection -** Classe responsável por atribuir um tag a cada pedido do Cliente e resposta do Servidor

- **Frame -** Classe responsável por encapsular uma mensagem entre o Cliente e o Servidor que é independente ao seu conteudo.

3 Descrição das Funcionalidades

No nosso sistema, o servidor tem a capacidade de receber várias mensagens do utilizador, sendo que a classe Servidor recebe-as e processa-as, associando a mensagem recebida a uma funcionalidade do sistema. As funcionalidades que o nosso sistema efetua serão explicadas de seguida:

1. No nosso sistema é necessário efetuar o registo ou a autenticação, como demonstra na figura 1, para posteriormente ser possível efetuar outras funcionalidades:

    - **Registar na Plataforma -** Esta operação equivale à opção 1 do menu de Login, onde um novo utilizador introduz o seu nome de utilizador e a sua password. Posteriormente, antes de registar na plataforma, verificamos se o nome do utilizador e a sua password já existem acendendo um Map e utilizando um lock. Este processo de Registo é implementado pelo método *adicionaNovoCliente* na classe classe GereClientes.
    - **Autenticar na Plataforma -** Esta operação equivale à opção 2 do menu de Login, onde o utilizador efetua a autenticação no servidor introduzindo o seu nome de utilizador e a sua password. Antes de autenticar na plataforma verificamos os dados acedendo um Map e utilizando um lock. Este processo de Autenticação é implementado pelo método *autenticaCliente* na classe GereClientes.

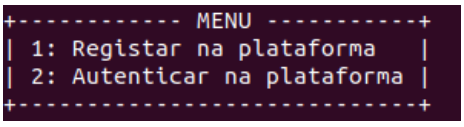


Figura 1: Interface inicial do Cliente.

2. Após o utilizador efetuar o registo ou a autenticação, as opções que mostramos na figura 2 aparecem na interface do utilizador possibilitando a execução de mais funcionalidades:

    - **Listar Trotinetes Livres -** Esta operação, corresponde à opção 1 do menu das operações e permite ao utilizador informar-se sobre as trotinetes livres no sistema próximas de uma dada posição.
    - **Listar Recompensas -** Esta operação, equivale à opção 2 do menu das operações e permite ao utilizador informar-se sobre as recompensas disponíveis próximas a uma posição dada pelo cliente.

- **Reservar Trotinete -** Esta operação, equivale à opção 3 do menu das operações e permite ao utilizador reservar uma trotinete que se encontre livre numa zona, na qual o cliente escolhe.
- **Estacionar Trotinete -** Esta operação, equivale à opção 4 do menu das operações e permite ao utilizador estacionar uma trotinete que tenha reservado anteriormente. Para estacionar, o cliente terá que fornecer o código associado à respetiva reserva e à posição onde pretende estacionar.
- **Notificações -** Esta operação, equivale à opção 5 do menu das operações e permite ao utilizador ativar as notificações das recompensas na àrea de uma dada posição, sendo que deste modo, o cliente é notificado sempre que uma recompensa aparecer. Além disto, o cliente possui a capacidade de desativar as notificações.
- **Logout -** Esta operação, equivale à opção 6 do menu das operações e permite encerrar a conexão do cliente com o servidor.

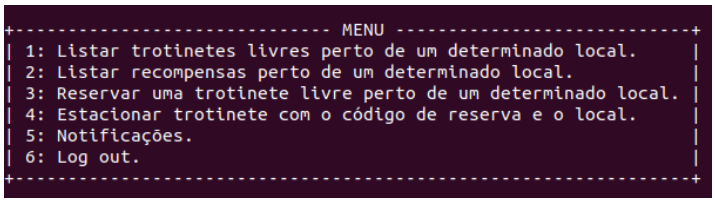


Figura 2: Interface do menu do Cliente.

4 Conclusão

Em modo de conclusão, após a finalização do trabalho prático da Unidade Curricular de Sístemas Distribuídos, sentimos que fomos capazes de aprofundar os nossos conhecimentos adquiridos nas aulas práticas de Sistemas Distribuídos, nomeadamente o uso de Locks e Conditions, que permitem que as zonas críticas sejam acedidas só por uma thread, não prejudicando assim a integridade e a consistência dos dados, além destes adquirimos conhecimento também sobre o paradigma Cliente e Servidor, ao utilizarmos sockets TCP.

Deste modo, finalizado o projeto, o grupo está satisfeito com o produto final, considerando ir de encontro com o que foi solicitado.