**Relatório Final de Sistemas Distribuídos – Grupo 8**



**Disciplina de Sistemas Distribuídos**

Ano Letivo de 2022/2023

Janeiro, 2023

**Diogo Neto - 8200435**

**Diogo Pinto – 8190173**

**Pedro Pereira - 8200431**

Conteúdo

[1. Introdução 3](#_Toc125132774)

[2. Aplicação 4](#_Toc125132775)

[2.1. Funcionamento 4](#_Toc125132776)

[2.2. Execução 5](#_Toc125132777)

[2.3. Implementação 5](#_Toc125132778)

[3. Conclusão 7](#_Toc125132779)

[4. Bibliografia 8](#_Toc125132780)

[Figura 1- Linhas ferroviárias em Portugal 3](file:///D:\Uni\3roAno\1roSemestre\SD\TP-SD\Report.docx#_Toc125129023)

[Figura 2- Serviços IP 4](file:///D:\Uni\3roAno\1roSemestre\SD\TP-SD\Report.docx#_Toc125129024)

# Introdução

Este documento serve como relatório do desenvolvimento do trabalho prático para a unidade curricular de Sistemas Distribuídos, o objetivo do qual era desenvolver uma aplicação que simulasse um sistema de notificações no contexto de uma rede ferroviária.

A fim de fazer o projeto aproximar-se mais da realidade, a rede ferroviária que foi tomada em consideração para este trabalho é constituída pelo conjunto de linhas em Portugal com tráfego ferroviário ativo com serviço de passageiros (Assinaladas a azul na figura 1).

Uma imagem com mapa

Descrição gerada automaticamente

Figura 1- Linhas ferroviárias em Portugal

# Aplicação

## Funcionamento

A aplicação divide-se em três partes principais. Cada uma dedica-se aos diferentes envolventes do sistema, sendo estes:

* Os passageiros;
* Os gestores locais;
* O gestor central.

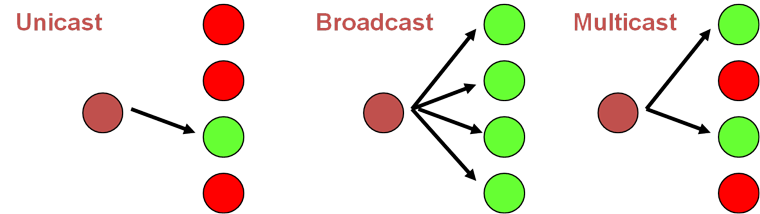
O funcionamento desta aplicação é baseado nas interações que estes três efetuam entre si, através dos serviços IP presentes na figura 2.

Figura - Serviços IP

Cada passageiro interage com o seu gestor local em *Unicast;* registando-se, autenticando-se e enviando notificações para este.

Os gestores locais interagem entre si e com o gestor central em *Multicast*.

O gestor central envia avisos de suspensão da rede ferroviária para todos os gestores locais em *Broadcast*.

## Execução

Comando para executar a aplicação do lado do Passageiro:

java -cp '../jar/json-simple-1.1.1.jar;.' passenger/Passenger <lineNumber> (1-24)

Comando para executar a aplicação do lado do Gestor Central:

java -cp '../jar/json-simple-1.1.1.jar;.' central/CentralManager

O servidor dos gestores locais é inicializado pelo IDE.

## Implementação

As interações dos envolventes da aplicação são efetuadas recorrendo a *sockets,* trocando-se mensagens em forma de objetos JSON, serializados pela biblioteca json-simple.

Quando necessário, é guardada informação em ficheiros JSON referente aos utilizadores, e às linhas. Estes ficheiros podem ser encontrados e examinados na pasta *files.* Apenas o servidor tem acesso a estes ficheiros.

A cada linha foi associada uma porta TCP -começando na 3000 e acabando na 3023- e um grupo *multicast*, sendo o IP do primeiro grupo ‘239.0.0.0’ e o do último ‘239.0.0.23’. É possível configurar o valor da primeira porta TCP, da porta e do endereço *multicast* através de variáveis na classe CentralManager.

A aplicação começa do lado do servidor (Server), que inicia 24 *threads* para cada gestor local (LocalManager); cada uma destas *threads* inicia duas *threads* imediatamente *-*uma para interações *unicast* (LocalManagerUnicastThread) e outra para interações *multicast* (LocalManagerMulticastThread)- e uma periodicamente, sendo esta dedicada ao envio de relatórios para o gestor central (ReportSenderThread).

A LocalManagerUnicastThread instancia uma ServerSocket e fica à espera de novas ligações. Com cada ligação recebida, é iniciada uma nova *thread* para lidar com as operações feitas em *unicast* com essa ligação (WorkerThread).

A LocalManagerMulticastThread é iniciada e fica à espera de receber notificações para as adicionar a um *array* de notificações.

A ReportSenderThread é iniciada a uma hora definida (neste trabalho foi definida as 18:00 como a hora de envio) se foram recebidas notificações e se um relatório ainda não foi enviado.

Do lado do servidor não existe interação de qualquer forma com a linha de comandos.

Do lado do passageiro, ao iniciar a aplicação, é efetuada uma ligação *unicast,* tendo agora uma WorkerThread associada. Daí, é possível autenticar ou registar-se.

No caso de se registar o passageiro é pedido para introduzir um nome de utilizador, uma palavra-passe e as linhas a que está associado. Caso haja qualquer complicação (nome de utilizador já existente, não escolhendo linhas nenhumas), o passageiro é pedido para introduzir estes campos outra vez. No caso de se autenticar, o passageiro é pedido para introduzir apenas o nome de utilizador e a palavra-passe.

Estas interações com a WorkerThread são possíveis através das classes PrintWriter (output) e BufferedReader (input).

Após se autenticar, a classe Passenger inicia *threads* para receber mensagens em *multicast* (MulticastThread), uma por cada linha a que esteja associado. Quando recebem uma notificação, estas imprimem um aviso para a consola. O passageiro consegue ver as notificações que recebe, ou enviar uma.

# Conclusão

# Bibliografia

Figura1-<https://www.dinheirovivo.pt/empresas/portugal-precisa-de-novas-linhas-para-entrar-nos-carris-13581313.html>

Figura 2-<https://www.kaankilic.net/unicastbroadcastmulticast-nedir/>