P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Disciplina de Sistemas Distribuídos

Ano Letivo de 2022/2023

Relatório Final de Sistemas Distribuídos – Grupo 8

Diogo Neto - 8200435

Diogo Pinto – 8190173

Pedro Pereira - 8200431

Janeiro, 2023

Conteúdo

1. 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).

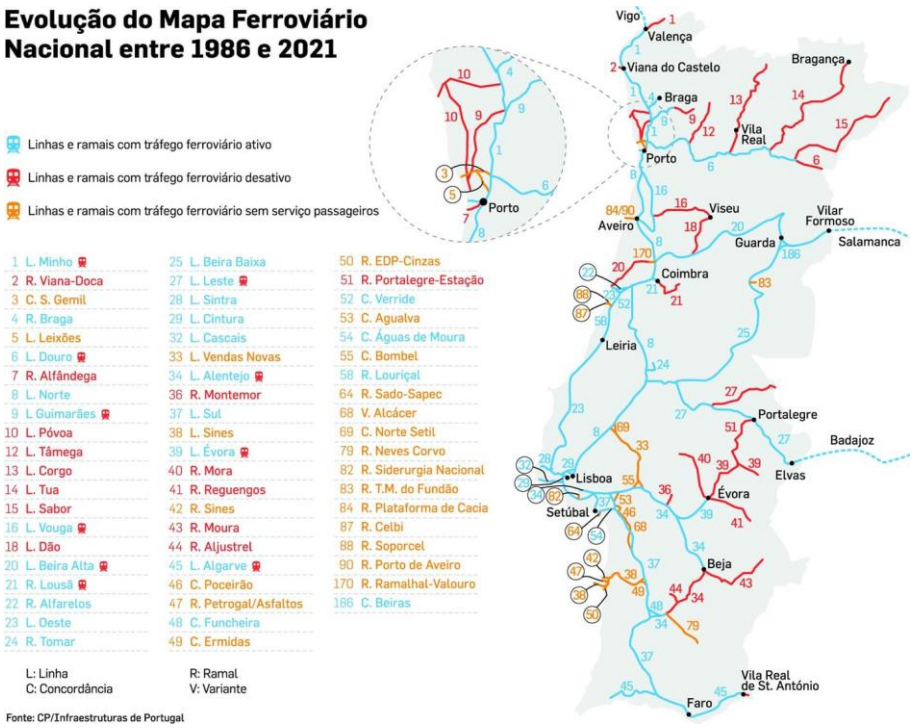

Figura 1- Linhas ferroviárias em Portugal

2. Aplicação

2.1. 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.

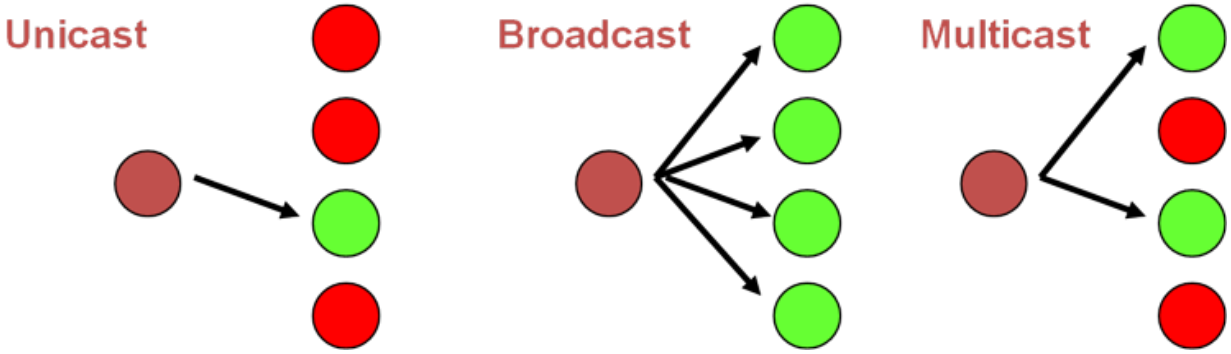

Figura 2- Serviços IP

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*.

2.2. 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.

2.3. 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 instância 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.

O servidor dispõe também de um AtomicReferenceArray para os *arrays* de notificações de cada gestor local. Este é atualizado cada vez que é enviada uma notificação e cada vez que um passageiro visualiza uma notificação.

Do lado do passageiro (Passenger), 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, 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). É gerado um objeto JSON com os campos fornecidos, cujos valores vão ser avaliados pela WorkerThread.
Após se autenticar, a classe Passenger inicia *threads* para receber mensagens em *multicast* (MulticastThread), uma por cada linha a que o passageiro está associado. Quando recebem uma notificação, estas imprimem um aviso para a consola. O passageiro consegue ver as notificações que recebe, ou enviar uma.
Ao enviar uma notificação, é pedido ao passageiro para introduzir as linhas que a vão receber e um comentário. É gerado um objeto JSON com os campos fornecidos e a data e hora a que foi enviada a notificação. O WorkerThread depois envia este objeto para os grupos associados às linhas escolhidas.
Ao ver as notificações, estas são impressas para a consola de uma forma formatada, para garantir que cada notificação aparece na consola uma -e apenas uma- vez.

Do lado do gestor central (CentralManager), ao iniciar a aplicação, é iniciada uma MulticastThread para a receção de relatórios do lado dos gestores locais. O IP do grupo a que esta *thread* está associada é o primeiro a seguir ao do último gestor (neste caso, ‘239.0.0.24’).
O funcionamento deste lado da aplicação é semelhante ao do lado do passageiro. É possível ver os relatórios de forma formatada, caso tenham sido enviados pelos gestores locais, e também é possível mandar uma notificação a toda rede ferroviária a dizer que o tráfego da rede foi suspenso.

Do lado do passageiro e do gestor central, interações por parte do utilizador são feitas com recurso à linha de comandos, do lado do servidor não existe interação de qualquer forma por parte do utilizador.

3. Conclusão

O desenvolvimento deste trabalho prático foi esforçoso, necessitando e aprofundando o nosso conhecimento sobre o funcionamento de *threads*, *sockets* a internet em geral.
A nossa maior lamentação em relação a este trabalho prático é a falta de uma interface gráfica devido à falta de tempo.

4. 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/