



ESCOLA  
SUPERIOR  
DE TECNOLOGIA  
E GESTÃO

# Sistemas Distribuídos

Licenciatura em Engenharia Informática

2022/2023

## Grupo 6

8200586, Bruno Ferreira

8200591, Nuno Castro

8200592, Jorge Correia

## Índice

<i>Introdução</i>	4
Opções estratégicas	5
Divisão em dois projetos	5
Armazenamento de informação em base de dados	6
Coleção Lines	7
Coleção ReportsLine	8
Coleção ReportsManager	9
Coleção Users	10
Encriptação da palavra passe	11
Acesso à base de dados	12
Relação entre gestores locais e linhas ferroviárias	12
Subscrição de linhas	12
Ambiente gráfico para o servidor	13
Fluxos do sistema	13
Passageiro	13
Gestor local	14
Servidor	14
Manual de compilação, configuração e utilização da aplicação	16
Ferramentas necessárias	16
Compilação	16
Configuração	17
Utilização	18
Autenticação	18
Servidor	20
Gestor local	22
Passageiro	23
<i>Tecnologias usadas</i>	26
Funcionalidades Implementadas	26
Descrição e justificação da utilização da matéria lecionada	27
Sockets TCP	27

Threads	27
Objetos partilhados	28
Arquitetura Cliente/Servidor	28
Enumeração das funcionalidades pedidas e não implementadas	29
<i>Conclusão</i>	30
<i>Bibliografia</i>	31
<i>Notas</i>	32
Link GitLab	32
Utilizadores já criados	32

## Introdução

No âmbito da unidade curricular de Sistemas Distribuídos foi-nos proposta a elaboração de um sistema de notificação de avisos sobre a alteração nos horários dos comboios. O principal objetivo deste sistema é então notificar todos os intervenientes envolvidos na rede ferroviária sobre alterações que tenham ocorrido em determinadas linhas.

Para a devida realização do trabalho, necessitamos de aplicar conhecimentos que foram lecionados durante as aulas da Unidade Curricular em questão, nomeadamente, comunicação entre sockets, gestão de threads, objetos partilhados, entre outros.

## Opções estratégicas

Para o desenvolvimento deste projeto foi necessário proceder à organização das ideias e à estruturação de uma solução que respondesse ao solicitado no enunciado fornecido. Tendo isto em conta, foi necessário tomar algumas decisões estratégicas que permitissem que o projeto fosse apresentado da forma pensada e com a devida qualidade técnica.

Por isso, iremos apresentar de seguida algumas das opções estratégicas tomadas e o respetivo motivo pelo qual foram tomadas.

### Divisão em dois projetos

Optamos por dividir a aplicação em dois projetos diferentes, sendo eles:

- **Cliente:** Projeto para a gestão de ações de passageiros e gestores locais
- **Servidor:** Gestão de conexões de clientes, gestão dos dados armazenados na base de dados e devido fornecimento de informação para os respectivos clientes conectados.

Existem diversas razões pela qual esta estratégia foi tomada, nomeadamente:

- **Modularidade:** Esta divisão fornece à aplicação um design mais modular, o que permite uma maior facilidade em mudar ou adicionar novos recursos a um componente sem afetar o outro.
- **Acoplamento fraco:** Ao separar em dois projetos diferentes, os detalhes de implementação são menos propensos a afetar o outro. Por isso, este esquema oferece um acoplamento fraco, ou seja, os componentes são independentes entre si o que torna a aplicação mais robusta e fácil de manter.
- **Separação de preocupações:** Ao possuir dois projetos diferentes permite que os desenvolvedores, neste caso os elementos do grupo, se concentrem em preocupações e problemas diferentes. Por exemplo, o desenvolvedor do cliente pode se concentrar mais na parte de criar uma interface intuitiva, enquanto que o desenvolvedor responsável pelo projeto do servidor pode se focar em realizar um back-end mais eficiente e escalável.
- **Reutilização:** A separação dos projetos permite que estes possam ser reutilizados em outros projetos sem a necessidade de re-implementar o outro componente.

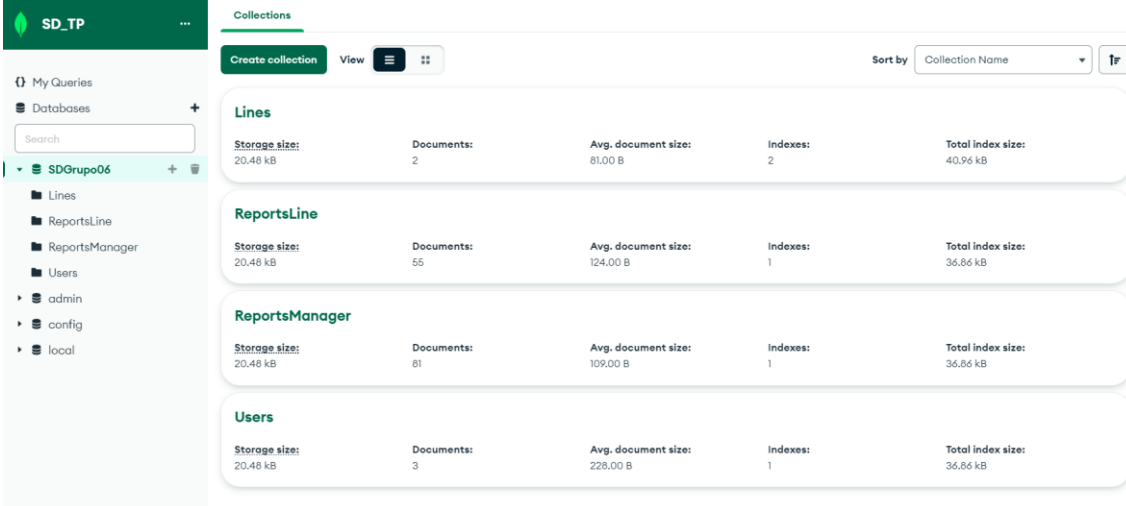
Tendo em conta os pontos explicados anteriormente, podemos concluir que possuir dois projetos diferentes torna a aplicação mais escalável e manutenível.

## Armazenamento de informação em base de dados

Apesar de não ser especificamente solicitado no enunciado o uso de uma base de dados para o armazenamento de informação, decidimos proceder à implementação de uma tendo em conta que seria a opção mais organizada e estruturada para o armazenamento de dados relacionados com o fluxo da aplicação.

Por isso, decidimos implementar uma base de dados em MongoDB, uma base de dados NoSQL o que permite que seja altamente escalável e flexível e que fornece inúmeras vantagens em relação a outros SGBDs (Sistemas de Gestão de Base de Dados), no contexto do projeto pedido.

Tendo isto em conta, de seguida será apresentada mais detalhadamente a base de dados criada e utilizada no âmbito da gestão de informação da aplicação.



Collection	Storage size	Documents	Avg. document size	Indexes	Total index size
Lines	20.48 kB	2	81.00 B	2	40.96 kB
ReportsLine	20.48 kB	55	124.00 B	1	36.86 kB
ReportsManager	20.48 kB	81	109.00 B	1	36.86 kB
Users	20.48 kB	3	228.00 B	1	36.86 kB

Inicialmente podemos observar, através do **MongoDB Compass** que existe uma base de dados chamada **SDGrupo06** que possui quatro coleções que serão também devidamente explicadas a seguir.

## Coleção Lines

Esta coleção refere-se ao local onde será armazenada toda a informação referente às linhas ferroviárias onde, tal como se pode observar na figura seguinte, possui o nome correspondente à linha, se a mesma se encontra ativa e as informações necessárias para a conexão do gestor local ao servidor de gestão da linha em questão (host e porta para a conexão).

```
_id: ObjectId('63b473a70cd9a4b013ab93d1')  
name: "Linha 1"  
isActive: true  
port: 2049  
host: "127.0.0.1"
```

```
_id: ObjectId('63b475970cd9a4b013ab93d2')  
name: "Linha 2"  
isActive: false  
port: 2048  
host: "127.0.0.1"
```

## Coleção ReportsLine

Esta coleção é responsável por armazenar informações referentes a avisos feitos por passageiros, gestores locais ou centrais. Nesta coleção os documentos armazenam a data e a hora a que o aviso foi enviado, o email do utilizador que procedeu ao aviso de alteração, o conteúdo da mensagem do aviso e a linha para que foi direcionado o aviso.

```
_id: ObjectId('63bda8de82eed811c6be31da')  
reportDate: "2023-01-10 18:05"  
userEmail: "manager1@gmail.com"  
message: "report teste"  
linha: "Linha 1"
```

```
_id: ObjectId('63bda8ee82eed811c6be31de')  
reportDate: "2023-01-10 18:05"  
userEmail: "client1@gmail.com"  
message: "report teste"  
linha: "Linha 1"
```

```
_id: ObjectId('63bda90f82eed811c6be31e2')  
reportDate: "2023-01-10 18:06"  
userEmail: ""  
message: "report"  
linha: "Linha 2"
```



## Coleção ReportsManager

A coleção ReportsManager possui a responsabilidade de armazenar a informação dos relatórios periódicos enviados pelos gestores locais de cada linha para o servidor central. Nesta coleção os documentos armazenam a data e a hora a que o relatório foi enviado, a linha para que foi feito o relatório, o número de aviso e de passageiros avisados.

```
_id: ObjectId('63b5da581fa0c07f654127a6')
relatorioDate: 2023-01-04T19:58:16.061+00:00
line: "Linha 1"
totalWarnings: 0
totalPassengersWarned: 0
```

```
_id: ObjectId('63b5db0c1fa0c07f654127a8')
relatorioDate: 2023-01-04T20:01:16.065+00:00
line: "Linha 1"
totalWarnings: 0
totalPassengersWarned: 0
```

```
_id: ObjectId('63b5dbc01fa0c07f654127aa')
relatorioDate: 2023-01-04T20:04:16.080+00:00
line: "Linha 1"
totalWarnings: 0
totalPassengersWarned: 0
```

## Coleção Users

Por fim, esta coleção armazena a informação referente aos utilizadores existentes, onde consta o nome, o email, a password encriptada, a role e a lista de linhas às quais o user está subscrito.

Caso o user seja um gestor local, apenas pode estar subscrito à linha que gere, enquanto que um mero passageiro pode estar subscrito às linhas que desejar.

```
_id: ObjectId('63b5d46f05cd343d5b43d430')
nome: "Manager"
email: "manager1@gmail.com"
password: "$argon2id$v=19$m=65536,t=10,p=1$XJW0aWiZEMzvt27L307WA$63u/bQ7PaTWZUH4..."
role: "Manager"
> lines: Array
```

```
_id: ObjectId('63b5d49605cd343d5b43d433')
nome: "Client"
email: "client1@gmail.com"
password: "$argon2id$v=19$m=65536,t=10,p=1$fB69mhce+GnmZn9VeufjAw$PCCSE/hBW5sCxRP..."
role: "Passenger"
> lines: Array
```

```
_id: ObjectId('63b5d5ecaff1de2c079c1d59')
nome: "Manager"
email: "manager2@gmail.com"
password: "$argon2id$v=19$m=65536,t=10,p=1$uKdoZBkpxXBNbkXr22jyzw$SFpTcoBYmtKqc/z..."
role: "Manager"
> lines: Array
```

## Encriptação da palavra passe

Decidimos que para o armazenamento das credenciais dos utilizadores da aplicação, a palavra-passe deveria ser devidamente protegida para que pessoas com acesso não autorizado não pudessem facilmente visualizar as credenciais e assim causar danos irreparáveis ao sistema. Para o efeito de encriptação demos então uso ao sistema de encriptação SHA (Secure Algorithm Hash).

Além de ser politicamente correto é também uma norma a cumprir segundo as regras estabelecidas no RGPD (Regulamento Geral de Proteção de Dados).

## Acesso à base de dados

Durante a organização e estruturação do projeto, decidimos que apenas o servidor teria acesso direto à base de dados.

Esta decisão foi tomada devido ao facto de ser uma boa prática e pelas seguintes razões apresentadas:

- **Validação de dados:** Ao centralizar a validação de dados do lado do servidor, pode-se garantir que todos os dados enviados pelos clientes são consistentes e atendem aos requisitos definidos na aplicação, antes de serem inseridos na base de dados.
- **Segurança:** Tendo em conta que apenas o servidor tem acesso à base de dados, torna-se mais fácil garantir que apenas utilizadores autorizados possam ler ou escrever dados na base de dados.
- **Consistência de dados:** Ao centralizar todo o acesso à base de dados através do servidor, garante-se que os dados sejam armazenados e recuperados de forma consistente, de maneira a que a integridade dos mesmos seja mantida.
- **Desempenho:** Como apenas o servidor tem interação com a base de dados, a quantidade de solicitações é muito mais reduzida, o que permite um desempenho muito mais elevado.

Por isso, se um passageiro ou um gestor local necessitar de ler ou escrever dados, estes terão de ser enviados para o servidor que, por sua vez, irá validar a operação e realizar ou não o pedido consoante o analisado.

## Relação entre gestores locais e linhas ferroviárias

Determinamos que uma linha apenas pode ser gerida por um gestor local, e este apenas pode gerir uma linha. Um gestor pode notificar os passageiros e o servidor sobre acontecimentos da linha e também pode fechar/abrir a linha.

## Subscrição de linhas

Um passageiro tem oportunidade de subscrever as linhas que pretender mas consequentemente, apenas recebe notificações dessas mesmas linhas a que está subscrito.

Para além disso decidimos que um passageiro tem acesso a uma página “Novas Notificações” onde tem acesso a notificações em tempo real de todas as linhas a que encontra subscrito, mas também tem acesso a uma página de histórico individual de cada linha.

## Ambiente gráfico para o servidor

Optamos por criar uma interface gráfica para o servidor para que este consiga gerir um conjunto de linhas ou até mesmo a rede completa, podendo assim enviar notificações referentes a uma ou mais linhas, notificando assim os respectivos gestores associados, que por sua vez irão avisar os passageiros subscritos.

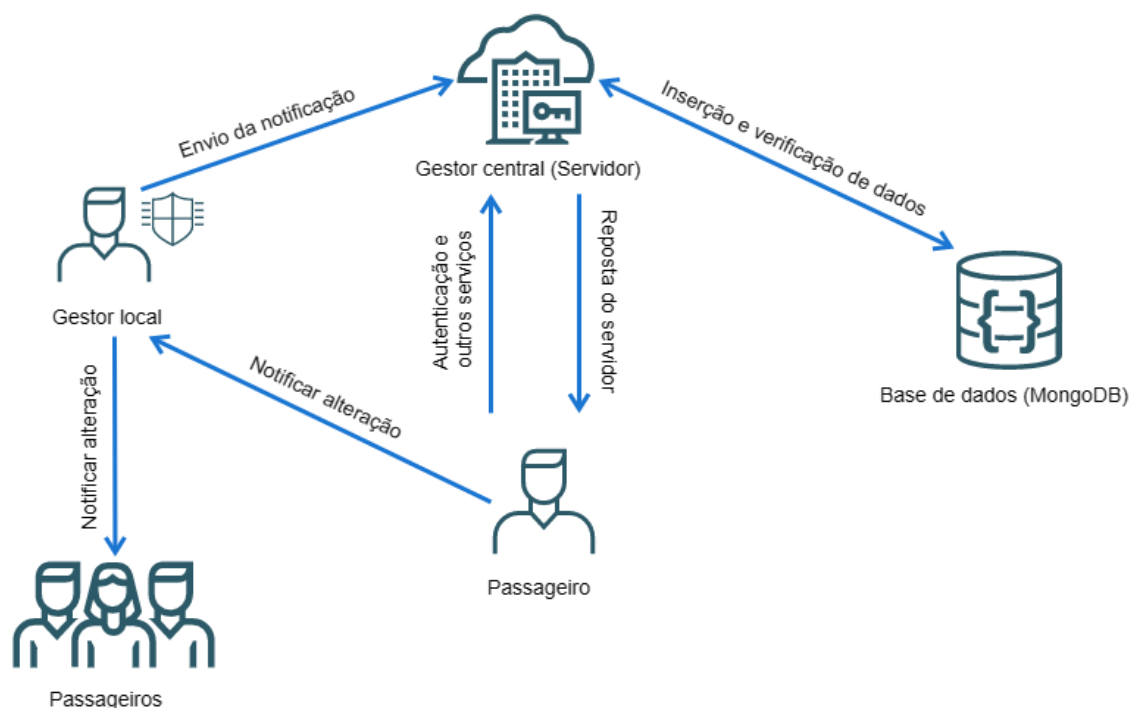
Podem também ser abertas/fechadas um conjunto de linhas ou até mesmo toda a rede de tráfego.

Além desta gestão de linhas, o servidor tem a responsabilidade de promover passageiros a gestores locais. Esta promoção remove a subscrição de todas as linhas subscritas enquanto este era passageiro, tendo em conta que este apenas poderá gerir uma linha.

## Fluxos do sistema

### Passageiro

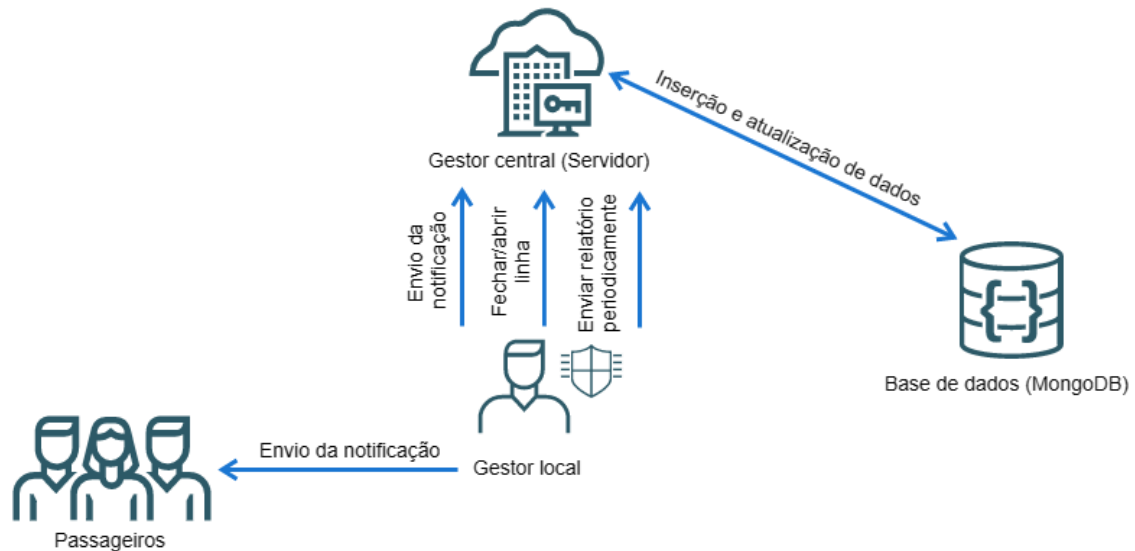
As notificações de um passageiro são enviadas para o gestor local da respetiva linha. Esse gestor recebe a linha, envia para todos os passageiros subscritos à mesma e por fim envia para o servidor para que este guarde a notificação na base de dados.



No diagrama os “outros serviços” correspondem à interação entre o utilizador e as suas linhas, ou seja, quando o utilizador subscreve ou cancela a subscrição de uma linha.

## Gestor local

As notificações do gestor local são enviadas para o servidor e para todos os passageiros da linha.

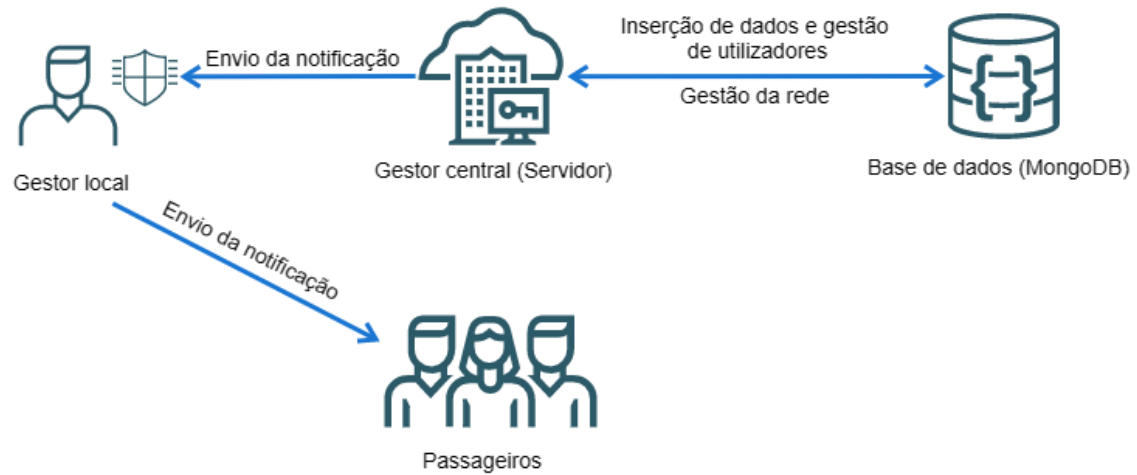


Além disso, o gestor faz a gestão da sua linha (abrir e fechar) comunicando com o servidor para atualizar a informação na base de dados.

Periodicamente é enviado para o servidor um relatório com o número de avisos e de passageiros notificados.

## Servidor

O servidor consegue enviar notificações para uma ou mais linhas, notificando os gestores locais associados a cada linha que por sua vez comunicam com os passageiros da sua linha.



O servidor tem a possibilidade de gerir as linhas da rede e os diversos utilizadores do sistema, podendo promover ou despromover cada um consoante necessário.

# Manual de compilação, configuração e utilização da aplicação

## Ferramentas necessárias

Para a execução do projeto realizado é necessário possuir minimamente as seguintes ferramentas:

- Java com a versão mínima 13
- Gradle com a versão mínima 7.5.1
- Docker com a versão mínima 20.10.20

## Compilação

Para proceder à devida compilação do projeto devem ser seguidos os próximos passos:

### 1. Ligar base de dados

Para ligar a base de dados basta se localizar na pasta Database e executar o comando `docker compose up`.

### 2. Ligar servidor

Para ligar o servidor basta se localizar na pasta "**`sd_grupo6_server`**" e executar o comando "**`gradle run`**".

### 3. Ligar a aplicação dos clientes e gestores

Para ligar a aplicação dos clientes e gestores basta se localizar na pasta `sd_grupo6_cliente` e executar o comando `gradle run`, para cada login que queira realizar.



## Configuração

Caso não deseje utilizar o mongodb através do docker, a base de dados pode ser importada localizando-se na pasta Database e executando um dos seguintes comandos:

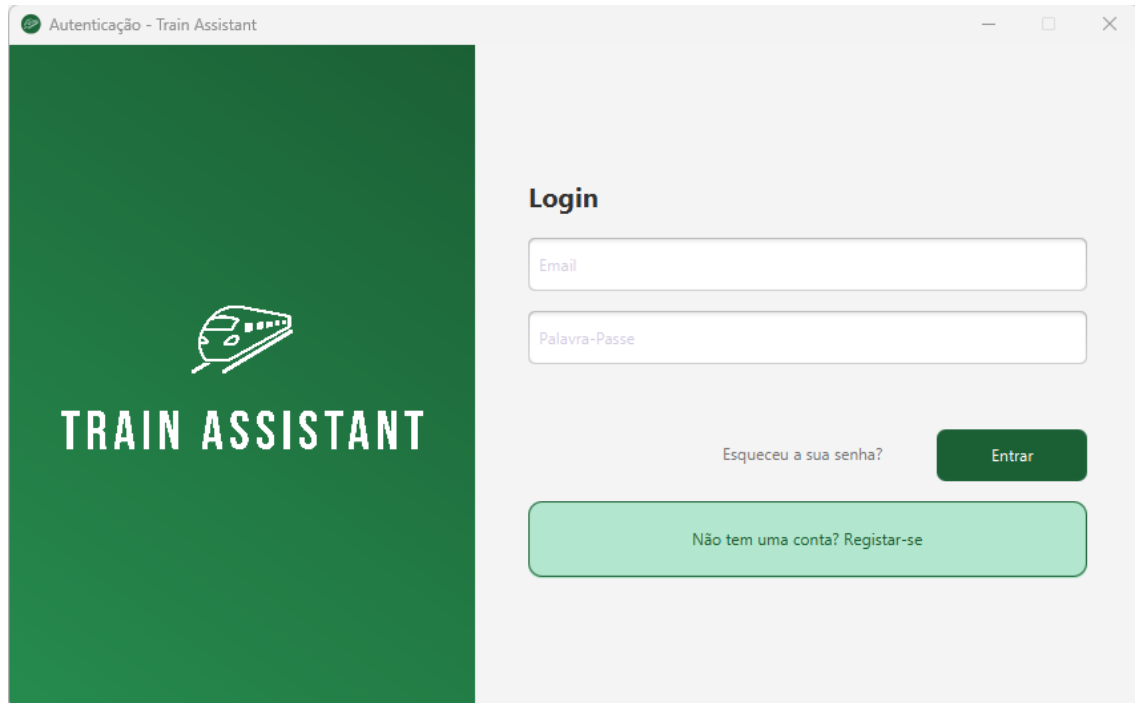
- `mongorestore --host host:port --username username --password password --authenticationDatabase admin --db database backup/database`
- `mongodump --uri "uri" --db database backup/database`

Após a execução de algum dos comandos apresentados anteriormente, a variável **MONGO\_URI**, localizada no ficheiro **.env** do projeto **sd\_grupo\_6\_server**, deve ser alterada para o URI da base de dados definida nos comandos anteriores.

## Utilização

### Autenticação

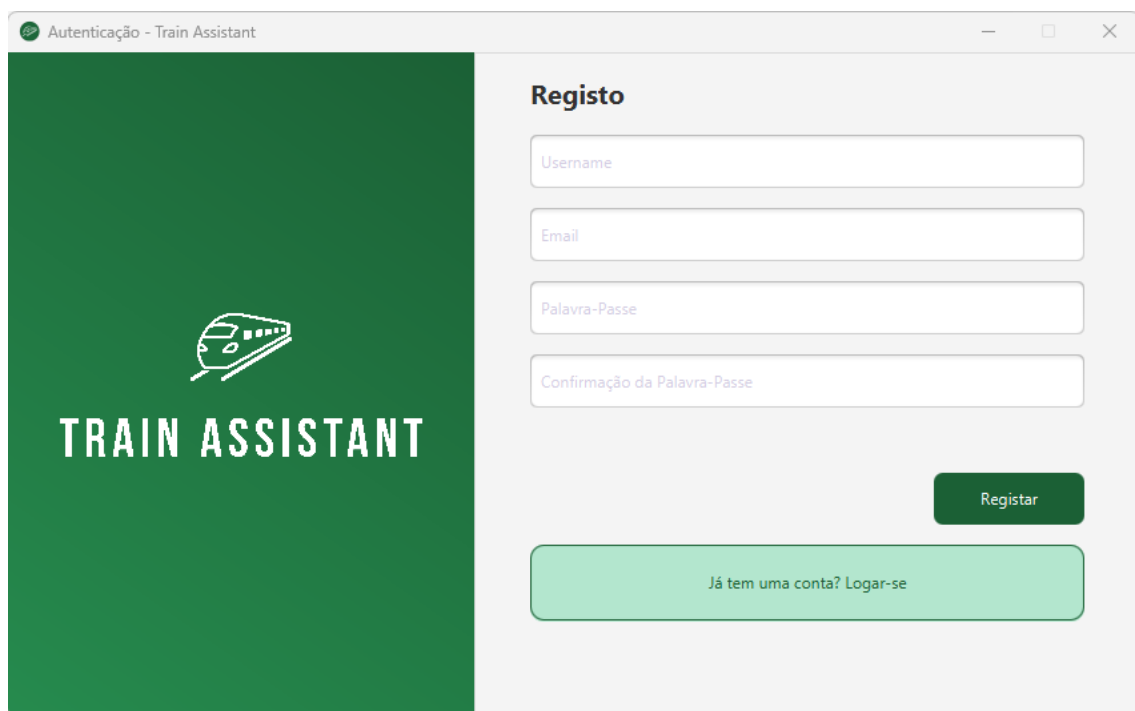
A página de login e registo são comuns aos passageiros e gestores locais:



The screenshot shows a web browser window titled "Autenticação - Train Assistant". The left side features a dark green background with a white train icon and the text "TRAIN ASSISTANT". The right side is a light gray area with the heading "Login". It contains two input fields: "Email" and "Palavra-Passe". Below these is a link "Esqueceu a sua senha?" and a dark green "Entrar" button. At the bottom, there is a light green button with the text "Não tem uma conta? Registrar-se".

Para efetuar login o utilizador necessita de inserir o seu email e password.

No final da página encontra-se o botão que permite ao utilizador registar uma conta caso este ainda não possua uma:



The screenshot shows a web browser window titled "Autenticação - Train Assistant". The left side features a dark green background with a white train icon and the text "TRAIN ASSISTANT". The right side is a light gray area with the heading "Registo". It contains four input fields: "Username", "Email", "Palavra-Passe", and "Confirmação da Palavra-Passe". Below these is a dark green "Registrar" button. At the bottom, there is a light green button with the text "Já tem uma conta? Logar-se".

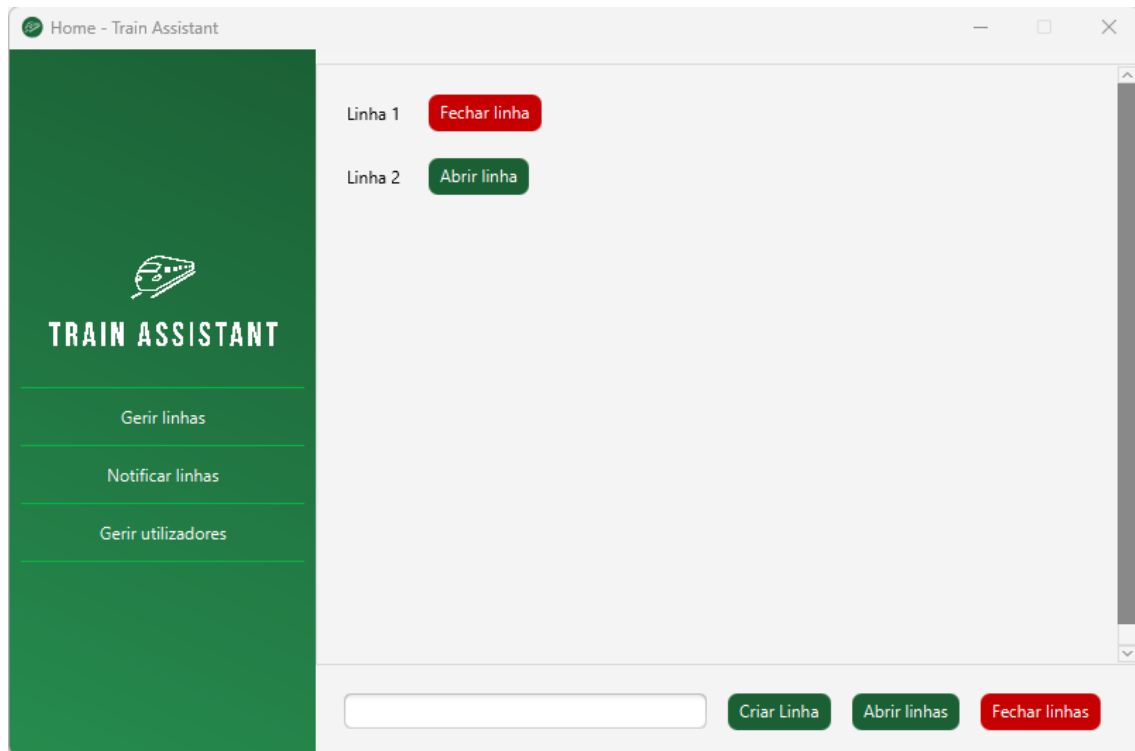
Para registar uma conta o utilizador necessita de inserir um username, o seu email, password e confirmação de password.

Tal como na página anterior, no final da página encontra-se o botão que permite ao utilizador efetuar login caso este já possua uma conta.

Utilizadores autenticados conseguem efetuar logout selecionando a opção “Sair” no menu lateral que é apresentado após este ser autenticado.

## Servidor

Ao abrir o servidor é apresentada a seguinte página(Gerir linhas):



Nesta página são apresentadas todas as linhas existentes no sistema e é possível gerir as mesmas ou até criar novas linhas.

Ao lado de cada linha existe um botão que permite abrir e fechar a respetiva linha.

No fundo da página é possível inserir o nome da linha e criar a mesma. Para além disso, através dos últimos dois botões, todas as linhas do sistema são fechadas ou abertas.

Ao clicar na opção “Notificar linhas” no menu que se encontra à esquerda, é apresentada a seguinte página:

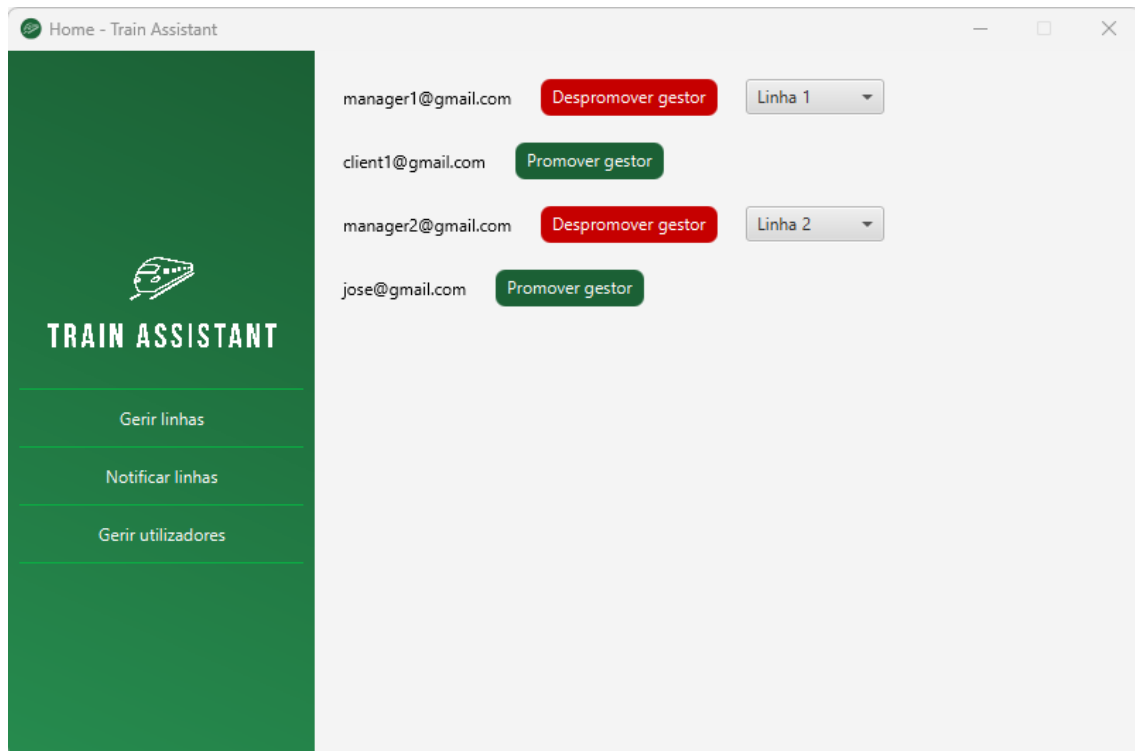
The screenshot shows a web browser window titled "Home - Train Assistant". The interface has a dark green sidebar on the left with the "TRAIN ASSISTANT" logo and three menu items: "Gerir linhas", "Notificar linhas", and "Gerir utilizadores". The main content area is light gray and displays a table with two rows of data:

2023-01-10T00:18:10.912644	Linha 1	report exemplo
2023-01-10T00:18:10.999347800	Linha 2	report exemplo

At the bottom of the main area, there is a form with a dropdown menu showing "Linha 1, Linha 2", a text input field containing "report exemplo", and a green "Enviar" button.

Esta página tem como objetivo notificar uma ou mais linhas (selecionadas na select box) com a mensagem inserida.

Ao seleccionar a opção “Gerir utilizadores” no menu, é apresentada a seguinte página:

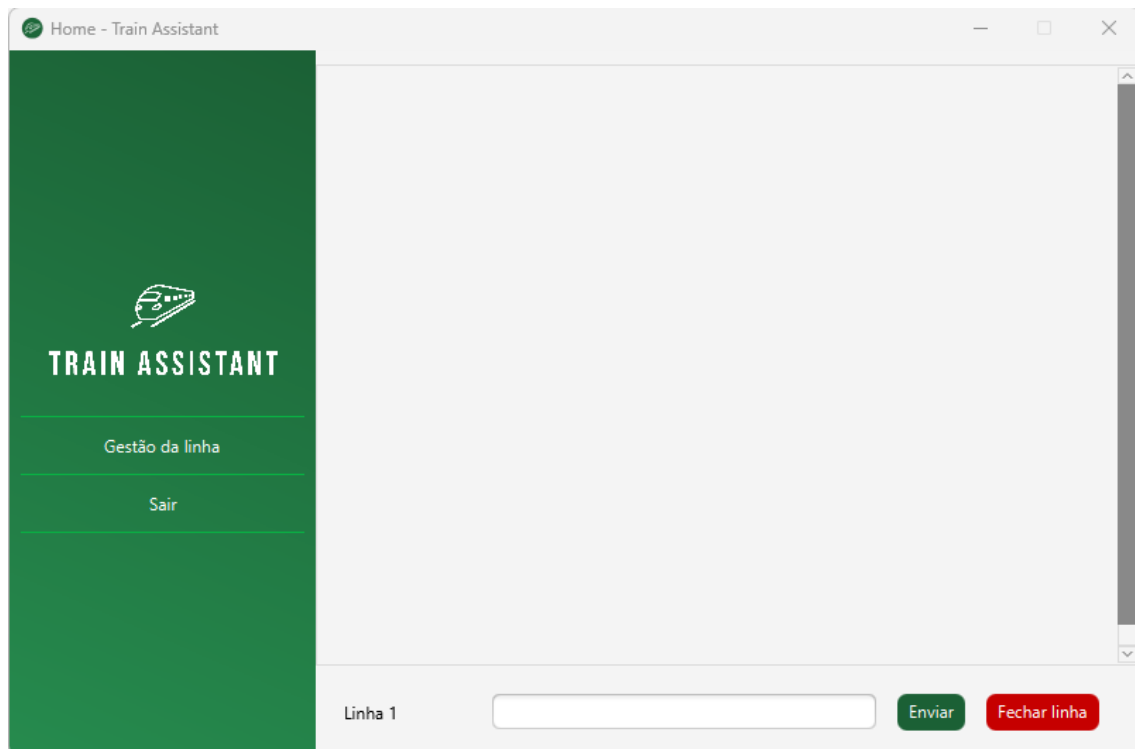


Através desta página é possível gerir os utilizadores do sistema. Um utilizador ao efetuar o registo na aplicação é registado como passageiro e pode ser promovido a gestor onde será necessário seleccionar a linha que este irá gerir.

Assim como um passageiro pode ser promovido, este também pode ser despromovido.

## Gestor local

Após finalizar a autenticação, será apresentada a seguinte página ao gestor local:

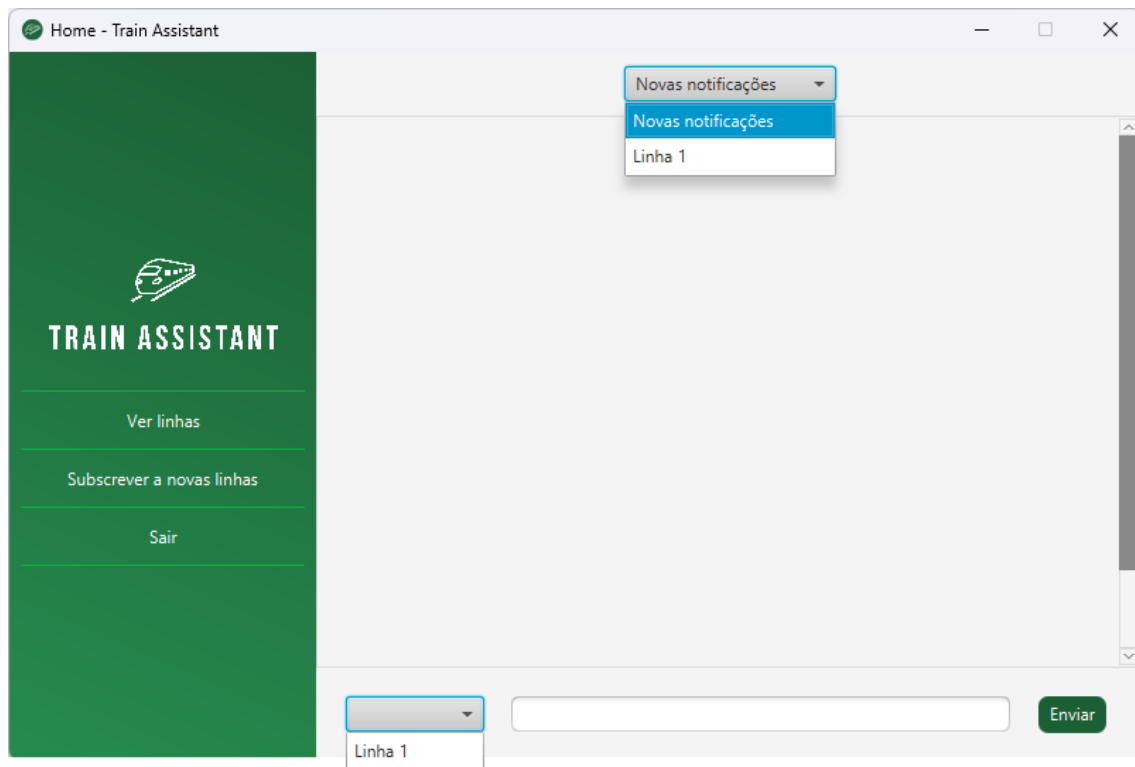


Através desta página o gestor local consegue notificar a linha que gere, enviando uma mensagem para todos os utilizadores que subscreveram a respectiva linha.

Para além de notificar, caso seja necessário, o gestor local pode fechar/abrir a sua linha.

## Passageiro

Após finalizar a autenticação, será apresentada a seguinte página ao passageiro(Ver linhas):



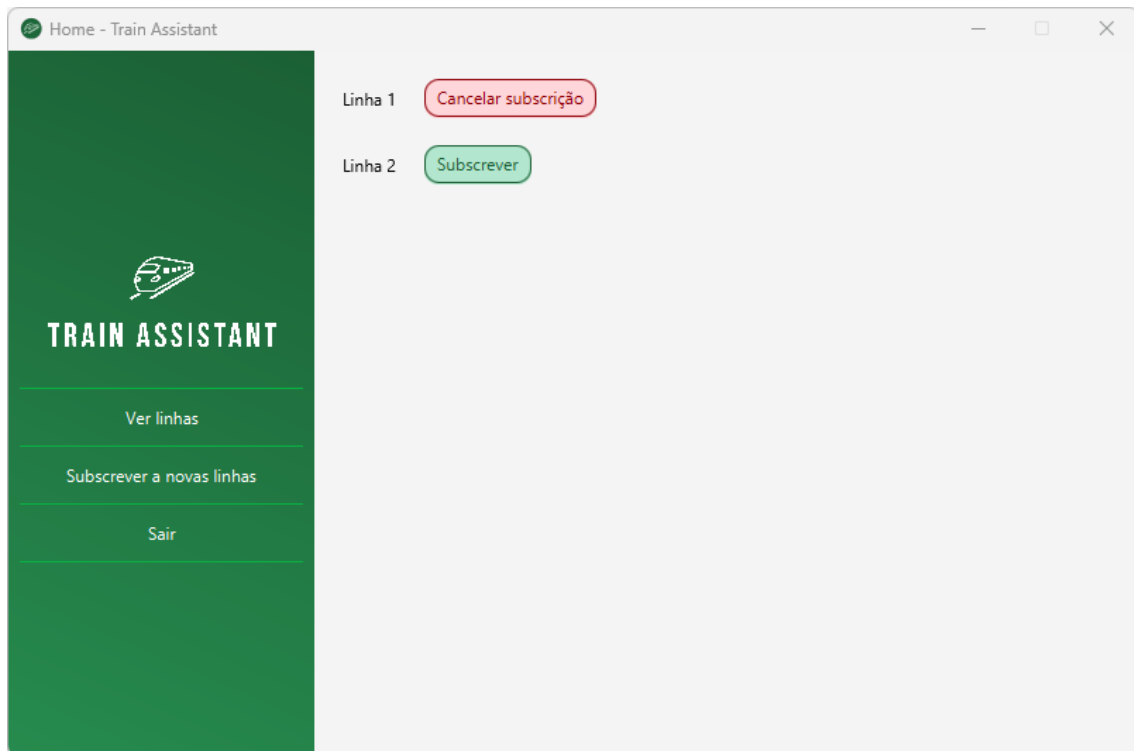
Caso a esteja seleccionada a opção “Novas notificações” na selectbox do início da página, o passageiro irá receber todas as notificações em tempo real das linhas a que está subscrito e onde exista um gestor local ativo.

Nessa select box, caso o passageiro selecione uma linha, este irá visualizar o histórico de notificações da respetiva linha.

No final da página, ao seleccionar uma linha na selectbox, o utilizador consegue notificar todos os utilizadores com acesso à linha.



Ao selecionar a opção “Subscrever a novas linhas” no menu lateral, é apresentada a seguinte página ao passageiro:



Nesta página o passageiro consegue visualizar todas as linhas existentes no sistema e para além disso, subscrever ou cancelar a subscrição das linhas, consoante deseje receber notificações.

## Tecnologias usadas

- Java 13
- Gradle 7.5.1
- MongoDB
- JavaFX
- Docker
- Git

## Funcionalidades Implementadas

- Autenticação
- Registo
- Clientes enviarem e receberem notificações das linhas que subscreveu
- Clientes subscreverem e cancelar a subscrição de linhas
- Gestores locais enviarem e receberem notificações da sua linha
- Gestores locais abrirem e fecharem a sua linha
- Gestores locais enviarem as notificações que recebem para os passageiros da sua linha
- Gestores enviarem de x em x tempo relatórios
- Servidor enviar e receber notificações da rede ou de n linhas
- Servidor suspender rede
- Servidor guardar histórico
- Servidor promover passageiros a gestores locais e vice-versa

## Descrição e justificação da utilização da matéria lecionada

Ao longo das aulas da unidade curricular de Sistemas Distribuídos foram lecionadas vários conteúdos que foram fundamentais na realização deste projeto.

De seguida, serão apresentados os diversos conteúdos utilizados e a sua importância e desempenho no âmbito do projeto desenvolvido.

### Sockets TCP

Os sockets foram utilizados neste projeto para realizar a comunicação entre os diferentes processos a correr na rede, nomeadamente:

- Entre passageiros e os gestores locais
- Entre os passageiros e o servidor
- Entre os gestores locais e o servidor

Utilizamos sockets baseados no protocolo TCP devido ao facto deste criar uma conexão virtual entre o emissor e o receptor antes de transmitir os dados, de modo a garantir que os dados transmitidos serão recebidos e na ordem correta, ao contrário do que acontece no protocolo UDP.

### Threads

Demos bastante uso a threads para podermos executar múltiplas funcionalidades em simultâneo, o que proporciona diversas vantagens, como:

- Melhoria do desempenho por permitir o processador executar diversas tarefas ao mesmo tempo
- Melhoria na resposta do sistema por executar várias tarefas em paralelo
- Melhoria na programação permitindo que as tarefas sejam divididas em blocos mais pequenos e, consequentemente, mais fáceis de gerir.

## Objetos partilhados

Objetos partilhados são importantes quando se trabalha com sockets, pois permitem que diferentes threads (ou seja, tarefas paralelas) acessem e modifiquem os mesmos dados de forma segura. Isso é especialmente importante quando se trabalha com sockets, pois as threads podem ser usadas para lidar com diferentes conexões de rede simultaneamente.

Sem objetos partilhados, as threads poderiam aceder e modificar os dados de forma inconsistente, o que poderia causar erros ou problemas de sincronização. Utilizando objetos partilhados, as threads podem aceder e modificar os dados de forma segura, garantindo que os dados sejam sempre consistentes e precisos.

Por estes motivos decidimos utilizar objetos partilhados para que a informação entre as diferentes screens não fosse inconsistente e assim trazer graves problemas ao funcionamento da aplicação.

## Arquitetura Cliente/Servidor

Tal como esperado, realizamos o trabalho baseado na arquitetura cliente/servidor em que, tal como já explicado anteriormente, cliente diz respeito aos passageiros e aos gestores locais enquanto que o servidor refere-se ao gestor central. Em aplicações que dão uso a esta arquitetura, o cliente é o dispositivo que inicia a comunicação e por sua vez faz os respectivos pedidos ao servidor, enquanto que o servidor é o responsável por responder a esses pedidos e fornecer os recursos solicitados.

## Enumeração das funcionalidades pedidas e não implementadas

Ao longo do projeto foram desenvolvidas todas as funcionalidades propostas no enunciado do trabalho prático, pelo que podemos concluir que não existem funcionalidades por implementar.

## Conclusão

O desenvolvimento deste projeto foi benéfico para o grupo dado que permitiu implementar os conhecimentos obtidos na UC. A resolução de problemas também é uma parte importante do projeto pois incentiva o trabalho em equipa de modo a chegar a soluções melhores.

No geral julgamos que o projeto foi desenvolvido com sucesso, existindo aspetos que poderiam ser melhorados em trabalho futuro.

.

## Bibliografia

- Conteúdos fornecidos no moodle
- [Stackoverflow](#)
- <https://www.mongodb.com/docs/drivers/java/sync/current/>

## Notas

### Link GitLab

[http://gitlab.estg.ipp.pt/sd\\_grupo6/sd\\_grupo6.git](http://gitlab.estg.ipp.pt/sd_grupo6/sd_grupo6.git)

### Utilizadores já criados

- Utilizadores do tipo passageiro (email:password)  
client1@gmail.com:Client123!
- Utilizadores do tipo gestor local (email:password)  
Manager1@gmail.com:Manager123!  
Manager2@gmail.com:Manager123!