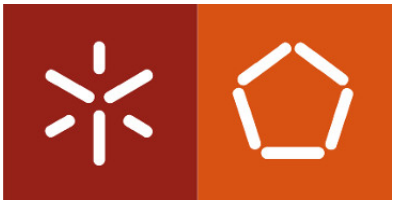


Universidade do Minho

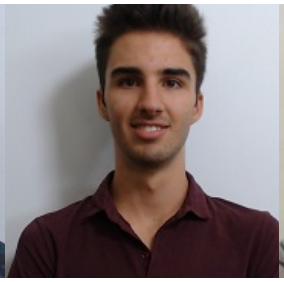
25 de janeiro de 2021

Relatório do trabalho prático - Grupo 29

Sistemas Distribuídos

Mestrado Integrado em Engenharia Informática - 3º ano






Adriano Novo Soto Maior	Bruno Pinto Jácome	José Pedro Ribeiro Peixoto	José Luís Abreu Mendes
A89483	A89515	A89602	A75481

1 | Contextualização

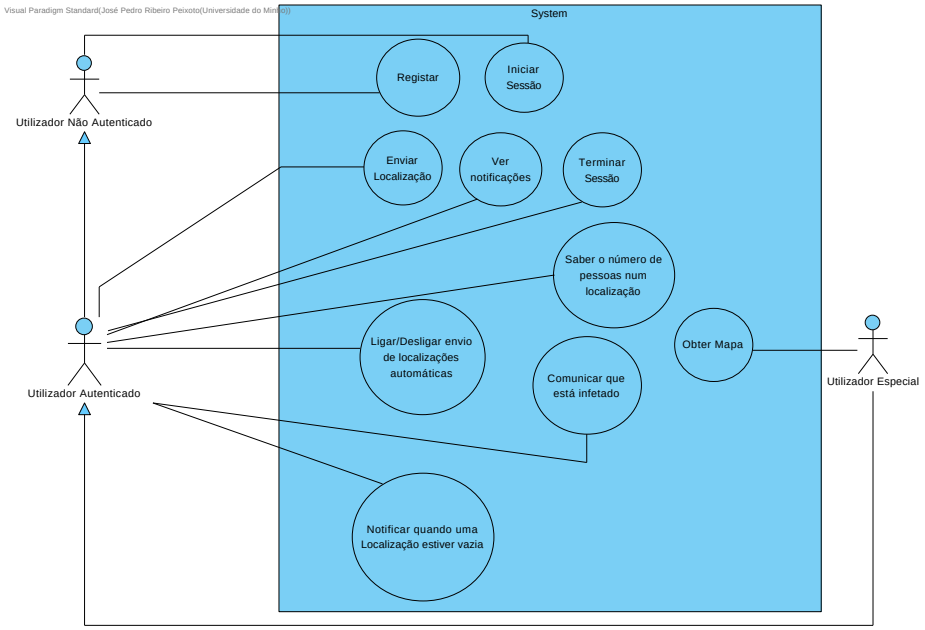
1.1 Introdução

Este relatório foi realizado no âmbito da Unidade Curricular de Sistemas Distribuídos, envolvendo o conceito de concorrência e de sistemas distribuídos, recorrendo à linguagem de programação **Java**. Tem como objetivos principais, abordar o raciocínio e mostrar a plataforma engenhada pelo grupo, que permite gerir e estudar a evolução da transmissão do CoViD-19 dos utilizadores dentro de uma área.
Este relatório divide-se essencialmente em 4 partes, todas relacionadas com a resolução do enunciado do trabalho prático: Na primeira parte, formula-se o problema e identificam-se todos os requisitos necessários para o funcionamento total do programa. Já na segunda parte, modela-se o problema e apresenta-se o diagrama de classes. Depois, analisam-se os resultados obtidos, demonstrando, em ação, as várias funcionalidades do programa. Em último lugar, apresenta-se uma breve conclusão acerca do projeto, das possíveis dificuldades encontradas e das futuras implementações.

1.2 Formulação do problema

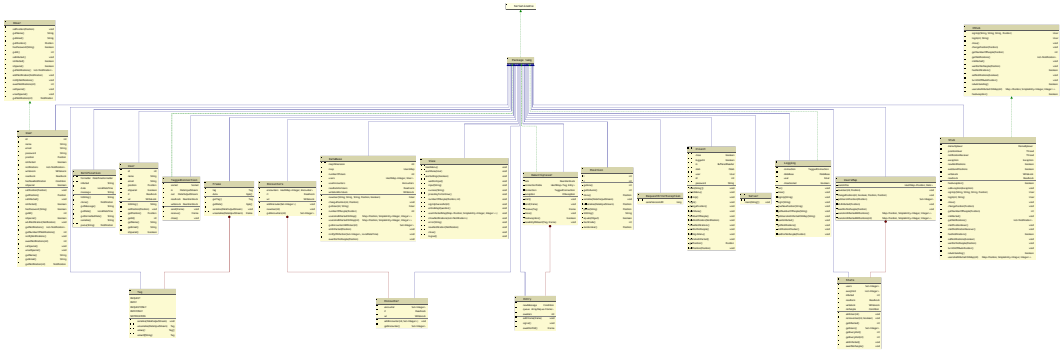
Neste projeto desconsiderou-se a privacidade de dados pessoais, tal como consta no enunciado do trabalho. Além disso, as localizações dos utilizadores são relativas a uma grelha de $N \times N$ locais, sendo as coordenadas geográficas pares discretos de índices. Além disso, quando um utilizador comunica que está infetado, a sua sessão termina e todos os utilizadores, que alguma vez tenham estado na mesma localização com este utilizador, são notificados.

1.2.1 Funcionalidades Básicas e Adicionais



2 | Programa

2.1 Diagrama de Classes

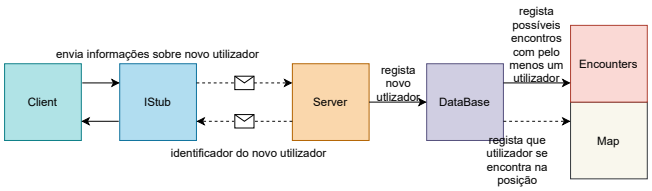


Abordagem: Em destaque estão três classes que são fundamentais no nosso programa:

- O **Demultiplexer** que permite que mais que uma *thread* espere por mensagens enviadas pelo servidor e até com a mesma *tag*.
- O **Logging** que é a *thread* responsável por toda a iteração com um utilizador, comunicando com o Stub do *Client*, desde o início até fim da sessão.
- O **Stub** que, embora organizada de uma forma incomum, estabelece uma conexão com o servidor e permite transparecer o acesso aos dados da **DataBase**.

2.2 Funcionalidades Básicas

2.2.1 Registo de um utilizador

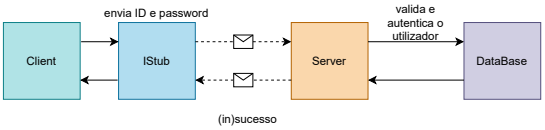



```
> 1
> Input name
> Pedro
> Input email
> pedro@gmail.com
> Input password
> segredo
> Input position : <line> <column>
> 1 1

The sign up was successful!
Your ID is: 1

ID: 1   Name: Pedro     Email: pedro@gmail.com  Position: 1 1   Special?: false
---------------------------------
```

Abordagem: Este método é bastante simples. Apenas se enviam as informações sobre o utilizador (nome, email, password, posição) para o servidor que são registadas na base de dados. A base de dados aloca espaço no mapa e na classe Encounters para poder registar a dinâmica

deste utilizador. Nessa classe Encounters, sempre que que se cria um utilizador, numa certa posição, regista-se um encontro entre os utilizadores que lá se encontram.
O servidor responde com o ID criado para este utilizador necessário para se voltar a autenticar.

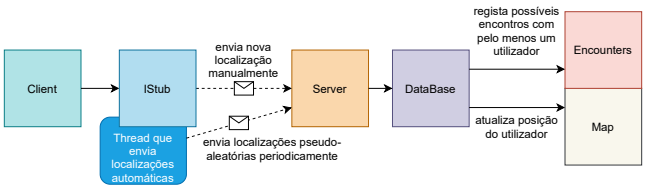
2.2.2 Autenticação de um utilizador




```
1. Sign Up
2. Log In
0. Sair

> 2
> Input id of the account
> 1
> Input password
> segredo
ID: 1   Name: Pedro     Email: pedro@gmail.com  Position: 1 1   Special?: false
-----------------------------------
```

Abordagem: Este método é novamente simples. Apenas se enviam o ID e a password do utilizador para o servidor. O servidor responde o (in)sucesso da autenticação.

2.2.3 Envio de localizações

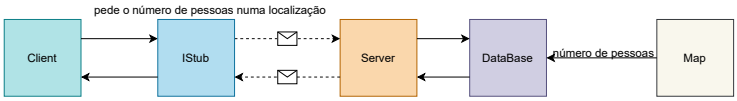



```
1. Give New Position
2. Get Number of People in a given Position
3. I'm infected
4. Show Notifications
5. Warn me when "No People in Position"
6. SETTINGS
0. Log Out

> 1
> Input new position: <line> <column>
> 2 2
```

```
1. Give New Position
2. Get Number of People in a given Position
3. I'm infected
4. Show Notifications
5. Warn me when "No People in Position"
6. SETTINGS
0. Log Out

> 6
1. Turn ON Auto-Send Positions
0. Go Back

> 1
1. Turn OFF Auto-Send Positions
0. Go Back
```

Abordagem: Novamente simples, mas obriga o cliente a estar autenticado. Há duas formas de enviar localizações (ou posições): Enviar manualmente a localização ou automaticamente através de um método que calcula uma posição pseudo-aleatória dentro dos limites do mapa. Em ambos os casos, o servidor não responde a este envie e simplesmente regista esta posição.

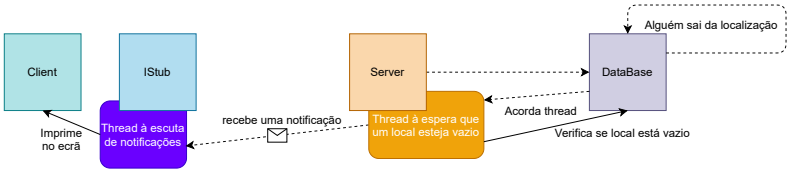
2.2.4 Número de pessoas numa dada localização



```
1. Give New Position
2. Get Number of People in a given Position
3. I'm infected
4. Show Notifications
5. Warn me when "No People in Position"
6. SETTINGS
0. Log Out

> 2
> Input position: <line> <column>
> 6 4
There is 1 person in 6 4
```

Abordagem: Mais uma vez é bastante simples e obriga, novamente, o cliente a estar autenticado. O utilizador envia a posição e o servidor responde com o número de utilizadores que se encontram lá no momento do pedido.

2.2.5 Notificar quando localização está livre

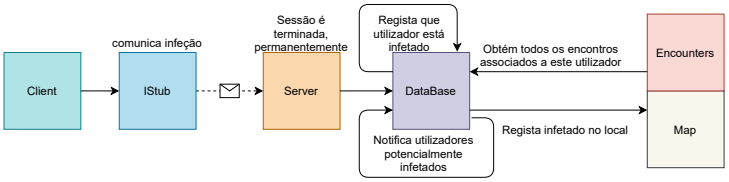



```
1. Give New Position
2. Get Number of People in a given Position
3. I'm infected
4. Show Notifications
5. Warn me when "No People in Position"
6. SETTINGS
0. Log Out

> 5
> Input position: <line> <column>
> 1 1
```

```
>>>  You have a new Notification:  <<<
> 2021-01-25 18:28 || The position 1 1 is currently empty. <
> Press to continue
```

Abordagem: Este método obriga o utilizador a estar autenticado. Para isto há 2 *threads* a correr no servidor em *background*: uma *thread* adormecida e que acorda sempre que novas notificações são enviadas ao utilizador e outra *thread* adormecida e que acorda sempre que utilizadores saiem da posição enviada pelo utilizador. Quando ninguém está na posição, esta última *thread* adiciona uma notificação ao utilizador, acordando a outra *thread* que posteriormente verifica a existência de uma nova notificação2.2.5, enviando-a pelo *socket* ao *client*.

2.2.6 Comunicar que utilizador está doente

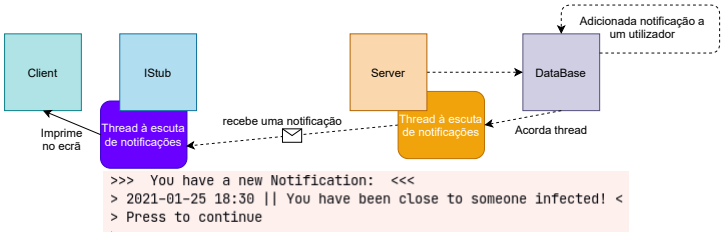



```
1. Give New Position
2. Get Number of People in a given Position
3. I'm infected
4. Show Notifications
5. Warn me when "No People in Position"
6. SETTINGS
0. Log Out

> 3

Logging out...
```

Abordagem: Este método obriga o utilizador a estar autenticado. O servidor regista que o utilizador está infetado e notifica todos os utilizadores, que alguma vez estiveram em contacto com ele, que estão potencialmente infetados. Isto será abordado com mais detalhe em 2.3.1. Além disso, o servidor termina, permanentemente, a sessão com este utilizador.

2.3 Funcionalidades Adicionais

2.3.1 Notificar utilizadores potencialmente infetados

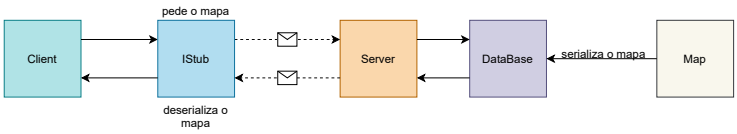


Abordagem:

- Quando se regista um novo utilizador num certa posição, todos os utilizadores que também estejam nessa posição têm um **Encounter** que é registado nos **Encounters**. Este encontro é registado para cada um dos utilizadores.
- Analogamente, quando um utilizador muda de localização volta-se a registar todos os encontros que ocorrem nessa sua nova localização.

Assim, sempre que alguém comunica que está doente, o servidor apenas observa os encontros associados a esse utilizador e notifica todos os utilizadores com quem ele se encontrou. Relativamente às notificações, assim que o *client* recebe uma mensagem do servidor com a *tag* **Notification**, é imprimido no ecrã a notificação recebida. Para isto, há uma *thread* à espera deste tipo de mensagens (*Notification*) e no momento em que alguma é recebida, a *thread* acorda.

2.3.2 Utilizador Especial: Descarregar mapa




```
7. Map of number of infected/users
0. Log Out

> 7

Print whole map? [Y/n]
>

The next map contains information of the number of infected users/total users in a certain position!

+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
|  000 / 001  |  001 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |
+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
|  001 / 000  |  003 / 000  |  003 / 000  |  001 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |
+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
|  002 / 000  |  003 / 000  |  000 / 000  |  001 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |
+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
|  000 / 000  |  004 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |
+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
|  003 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |
+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
|  001 / 000  |  001 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  001 / 000  |  003 / 000  |  000 / 000  |  000 / 000  |
+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
|  000 / 000  |  002 / 000  |  000 / 000  |  000 / 000  |  002 / 000  |  001 / 000  |  003 / 000  |  004 / 000  |  000 / 000  |  000 / 000  |
+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
|  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  001 / 000  |  003 / 000  |  002 / 000  |  000 / 000  |
+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
|  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  001 / 000  |  001 / 000  |  001 / 000  |  000 / 000  |  000 / 000  |
+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+
|  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |  000 / 000  |
+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+-----------+

> Press to continue
>
```

Abordagem: Este método está apenas disponível para utilizadores especiais autenticados. Há duas opções para isto, ou descarrega-se o mapa completo acerca de todas as visitas de todos os utilizadores ou descarrega-se o mapa de todas as visitas de um dado utilizador. Em ambos os casos o servidor envia-se o mapa pedido e o *client* imprime-o no ecrã. Nota: As visitas de um utilizador que mais tarde comunica que está doente, são ”convertidas”em visitas de um doente.

3 | Conclusão

3.1 Problemas de escala

Número de threads no Server

Para cada conexão estabelecida com um utilizador é criada uma thread, que se responsabiliza de toda a comunicação com o utilizador, uma thread "à escuta"de novas notificações do utilizador e, por último, uma thread por cada local que o utilizador peça para o alertar assim que esteja vazio. Sendo o sistema centralizado, por cada utilizador vão estar no máximo $1+1+N^2$ threads e é fácil de ver que o servidor não aguentará com um elevado volume de sessões ativas.

3.2 Futuras Implementações

Tempo limite como Potencialmente infetado

Considerando o modelo atual, os encontros entre utilizadores estão em memória indefinidamente. Contudo, os encontros com utilizadores infetados poderiam ser desprezados após 14 dias[1] desde o último encontro com um utilizador infetado.

Ficheiro de log do servidor

Atualmente, o servidor não regista as ações de utilizadores, inclusive eventuais erros no processamento dos comandos. Registar estes eventos facilitaria e acelararia o processo de *debugging* e, consequentemente, o serviço oferecido aos utilizadores iria evoluir mais rapidamente e mais eficazmente.

Base de Dados

Entendeu-se que um dos objetivos deste projeto seria o de exercitar o pensamento crítico sobre problemas de concorrência, nomeadamente no registo e gestão de dados e, por isso, criou-se uma classe simuladora de uma base de dados. No entanto, o próximo passo seria migrar as informações sobre utilizadores, encontros e o mapa, para uma base de dados robusta (p.e. MYSQL), que permitisse registá-las duradouramente.

GUI

Embora este trabalho não se focasse na criação de uma vista gráfica da aplicação, esta tornaria o acesso mais prático tanto para os utilizadores como para a equipa técnica, além de oferecer um serviço mais apelativo.

3.3 Resultado final

Em geral, considerando os objetivos deste projeto, estamos bastante satisfeitos com o resultado final da aplicação. Conseguimos implementar tanto os requisitos básicos como os adicionais. Todavia, poderiam ter sido adotadas abordagens diferentes durante o desenvolvimento da plataforma. Por exemplo, neste caso, criaram-se notificações genéricas que serviram para notificar utilizadores sobre um contacto com alguém infetado e para notificar utilizadores que uma dada localização está vazia. Obviamente que não havia necessidade explícita desta implementação, mas pareceu ao grupo o mais realista e prático para um utilizador.

[1] `https://www.sns24.gov.pt/tema/doencas-infecciosas/covid-19/#sec-5`