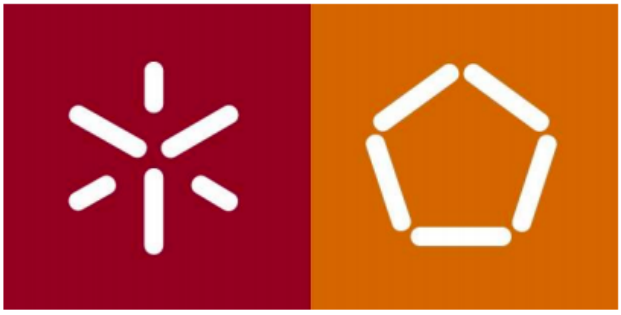

Universidade do Minho

Mestrado Integrado em Engenharia Informática

Paradigmas de Sistemas Distribuídos

Alarme Covid

Eduardo Lourenço da Conceição (A83870)
Ricardo Filipe Dantas Costa (A85851)
Rui Nuno Borges Cruz Oliveira (A83610)
Cândido Filipe Lima do Vale (PG42816)

18/01/2021
Braga

Índice

1 Introdução

Para o trabalho de Paradigmas de Sistemas Distribuídos, foi-nos proposta a construção de uma plataforma para suporte de rastreio de contactos e deteção de concentração de pessoas, envolvendo várias componentes de software: o cliente, o servidor de front-end, os servidores distritais e o diretório.

Neste serviço, os clientes ligam-se ao servidor front-end, que, após autenticá-los, recebe pedidos e reencaminha-os para um servidor distrital, conforme o distrito dos clientes. Estes servidores distritais, por sua vez, processam os pedidos enviados pelos clientes, sendo que estes, além de requisitar serviços, podem também ser notificados relativamente a informação que achem pertinente. Por fim, temos o diretório, que disponibiliza uma interface *RESTful*, que pode ser utilizada para consulta estatística de informações do sistema.

Os componentes de softwares referidos deverão ser implementados utilizando Java para clientes, servidores distritais e diretório, Erlang para servidor de front-end e ainda ZeroMQ e Dropwizard.

2 Arquitetura do sistema

Inicialmente, começamos por estruturar a solução do sistema, de modo a simplificar a sua implementação e corresponder com os requisitos pedidos. Através da seguinte figura, conseguimos analisar o funcionamento geral do sistema.

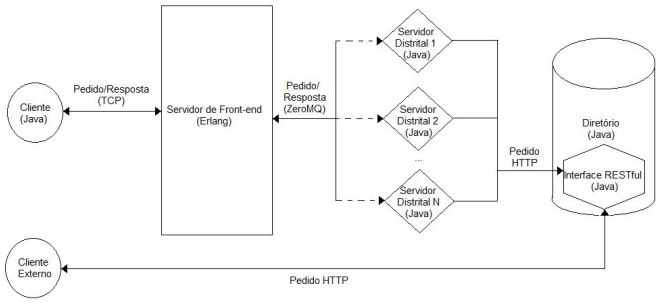

Figure 1: Arquitetura geral do sistema

Relativamente à comunicação entre os diferentes componentes escolhemos por formato de texto, visto que optamos por não prescindir da legibilidade.

No que diz respeito à propagação dos pedidos entre as diferentes componentes da arquitetura, a estratégia utilizada foi a seguinte: a interação entre o cliente e o servidor front-end é feita com sockets *TCP* para distinguir as diferentes sessões dos clientes. Por sua vez, o servidor de front-end e os servidores distritais comunicam através de sockets *ZeroMQ REQ-REP* para enviar e tratar os pedido, devolvendo a respetiva resposta. Relativamente às notificações, recorremos a sockets *ZeroMQ*

PUB-SUB de modo a expressar interesse por parte dos clientes em receber determinadas notificações, ou seja, subscrever determinados eventos segundo um critério específico. Por último, os servidores distritais vão atualizando a informação no diretório existente, efetuando pedidos *HTTP*, através da interface *RESTful* disponibilizada pelo mesmo.

3 Implementação

3.1 Estratégias adotadas

Antes da implementação propriamente dita, precisamos de definir algumas estratégias sobre pontos chaves do nosso sistema, como a divisão dos servidores distritais e as localizações dos utilizadores.

Relativamante aos servidores distritais, admitimos que correspondem a 18 distritos, mais propriamente, os 18 distritos respetivos ao país Portugal. Cada um destes forma uma grelha de NxN locais(o N escolhido foi 5), sendo as coordenadas geográficas pares discretos de índices.

No que diz respeito à localização dos utilizadores, assumindo que cada um reside num distrito e não se movimenta entre distritos, a sua localização é alterada apenas quando efetua login, sendo esta um valor aletório dos pares de índices.

3.2 Cliente

O cliente envia e recebe mensagens ao servidor front-end através de sockets *TCP*, como já referido. Este tem ao seu dispor uma interface, que lhe permite, inicialmente, registar-se e conectar-se ao sistema. O servidor front-end efetua a autenticação do cliente, retornando uma resposta positiva ou negativa conforme a validade dos dados. Após estar autenticado, o cliente tem ao seu dispor o menu seguinte:

1. Obter o número de pessoas numa dada localização do distrito,
2. Comunicar que se encontra doente,
3. Pretender registar/cancelar o interesse em receber notificações,
4. Logout.

3.2.1 Notificações

Relativamente às notificações, ou seja, o terceiro ponto, é criado no cliente uma socket *ZeroMQ SUB* e é lançada uma thread, isto é, o **Subscriber**. Esta thread está a escuta de publicações que vão sendo feitas, notificando o cliente quando elas chegam.

Sempre que um utilizador é autenticado e entra no sistema, são feitas duas subscrições automáticas, sendo que a primeira é relativa aos utilizadores com quem

já contactou anteriormente noutras sessões(caso não seja a primeira vez que acede ao sistema) de modo a ser notificado caso estes comuniquem que estão infetados. O tópico utilizado de subscrição para esta notificação foi o seguinte: **infetado:utilizador**.

A segunda subscrição automática é relativa à sua localização e distrito, uma vez que após estar no sistema, alguém pode aceder depois ao mesmo, com a mesma localização e distrito, e comunicar que está doente. Para esta segunda subscrição, foi utilizado o seguinte tópico: **infetado:distrito:localizacao**.

Para além destas, o cliente pode optar por ativar ou desativar notificações, mais propriamente as seguintes, acompanhadas respetivamente pelos tópicos usados:

1. Notificação quando deixar de haver pessoas numa dada localização de um dado distrito, **saida:distrito:localizacao**
2. Notificação sobre o aumento da concentração de pessoas numa dada localização de um dado distrito, **aumento:distrito:localizacao**
3. Notificar acerca da diminuição da concentração de pessoas numa dada localização de um dado distrito, **dimuicao:distrito:localizacao**
4. Notificar em relação à ocorrência de mais um infetado num dado distrito, **ocorrencia:disrito**

3.3 Servidor de front-end

Como já referido, esta componente é a ponte entre os clientes e os servidores distritais, tendo sido desenvolvida em Erlang. Além disso, foi também utilizada ZeroMQ para as sockets que permitem a conexão aos servidores distritais. Este encontra-se responsável, por um lado, pela autenticação de clientes e encaminhamento dos seus pedidos, e, por outro lado, por intermediar notificações.

3.3.1 Autenticação

O servidor mantém as informações dos utilizadores atualizadas num *map*. Como tal, sempre que alguém é registado com sucesso, os seus dados são adicionados ao *map* de utilizadores que o sevidor mantém. Por outro lado, quando é efetuado um login, os dados são validados com base na informação do mesmo *map*.

Cada utilizador tem associado como key o seu username e como values a sua password, um booleano respetivo ao seu login(true ou false, caso esteja logado ou não, respetivamente),o seu distrito e o seu estado(saudável ou doente).

3.3.2 Encaminhamento de pedidos

O front-end faz chegar aos servidores distritais os pedidos de cada cliente, com base no seu distrito. Logo, temos 18 servidores independentes a tratar dos pedidos dos clientes, o que permite uma maior escalabilidade ao sistema.

3.3.3 Intermediário de notificações

Esta componente é também responsável por intermediar a propagação de notificações provenientes dos servidores distritais, de modo a distribuir o custo de propagação das mesmas. Posto isto, o servidor de front-end funciona também como *broker*, sendo lançado um processo que irá agir como proxy. Este processo tem na sua posse sockets *PUB-SUB*, em que o segundo subscreve tópicos, recebendo notificações dos servidores distritais, e o primeiro encaminha-as para os clientes conectados.

3.4 Servidor Distrital

Os servidores distritais são responsáveis por receber do front-end os pedidos dos clientes, através de um socket ZeroMQ *REP*, tratando cada um de forma adequada. Cada um deles mantém o registo dos utilizadores, infetados e possíveis infetados.

Para tratar dos pedidos, os servidores distritais mantém um histórico dos utilizadores que estiveram nas suas localizações, organizado sob a forma de um *map* de *listas*. Este histórico é atualizado sempre que um cliente sai, através da inserção de uma nova *lista* no histórico, *lista* esta composta pela última existente no *map*, removendo o cliente que sai. Por outro lado, é também atualizado quando um cliente entra, caso a localização em que o cliente se encontra tenha histórico é adicionado à última *lista* do mesmo, caso contrário cria uma nova com este cliente e adiciona ao histórico.

Os pedidos dos clientes podem ser dois(além da manter a informação relativa às suas localizações), obter o número de pessoas numa dada localização do seu distrito e comunicar que está doente.

3.4.1 Obtenção do número de pessoas numa dada localização

Caso um cliente pretenda aceder a esta informação, o servidor distrital, através do histórico da localização dada que mantém, acede ao mesmo e devolve o tamanho da última *lista* do *map*, visto que é a atual.

3.4.2 Comunicação de infeção

No caso de um cliente comunicar que se encontra doente, o servidor distrital envia a notificação de que este esteve infetado e todos aqueles que estiveram em contacto com o mesmo receberão a notificação.

3.4.3 Notificações

Uma vez que são os servidores distritais que tratam de processar os pedidos dos clientes é nestes que serão enviadas as publicações relativas à saída ou entrada de utilizadores no sistema, comunicação de utilizador doente e também de dimuição

ou concentração de utilizadores numa dada localização. Para tal, é criado um **Publisher**, partilhado por todos os servidores distritais, que é responsável por enviar todas as mensagens correspondentes aos tópicos referidos inicialmente no ponto 3.2.1.

3.4.4 Atualização do diretório

Quando um servidor distrital recebe a informação da localização de um dado cliente, este efetua um pedido HTTP POST para que o utilizador seja inserido na estrutura do diretórtio. Para além disto, sempre que um cliente comunica que está doente, faz dois pedidos HTTP POST, para que o infetado seja inserido no diretório e outro para inserir os possíveis infetados. Este processo é concretizado reccorendo à biblioteca Jackson e ao Apache HTTP Client.

3.5 Diretório

Esta componente funciona como um repositório de dados estatísticos relativos à app Alarme Covid. Apesar de ter sido implementado em Java, com o auxílio da *framework* Dropwizard, disponibiliza também uma interface *RESTful*, que pode ser acedida por clientes externos através de pedidos HTTP GET, sendo que para realizar o parsing do JSON, utilizamos a bibilioteca Jackson.

A informação posível de consultar é a seguinte:

- número de utilizadores e infetados de um dado distrito(/diretorio/distritos/nome)
- top 5 dos distritos com maior rácio de infectados/utilizadores e das localizações que tiveram o maior número de pessoas em simultâneo(diretorio/top5)
- número médio de utilizadores que se cruzaram com utilizadores declarados doentes(diretorio/medio)

Para além destas consultas, a interface do diretório disponibiliza também formas de atualizar a informação nele contida. Os componentes responsáveis por estas atualizações são os servidores distritais, tal como explicado em 3.4.4.

4 Conclusão

Em geral, acreditamos ter cumprido todos requisitos do trabalho, sendo que foram exploradas ainda outras funcionalidades, como, por exemplo, a criação de um broker para distribuir os custos de propagação das notificações.

Futuramente, poderia ser melhorada a interface com o utilizador, apresentando detalhes como valores numéricos a auxiliar o top 5. Para além disto, poderiam ser adicionados mais intermediários, ou seja, *brokers*, de modo a tornar o processo de routing das notificações ainda mais eficiente.