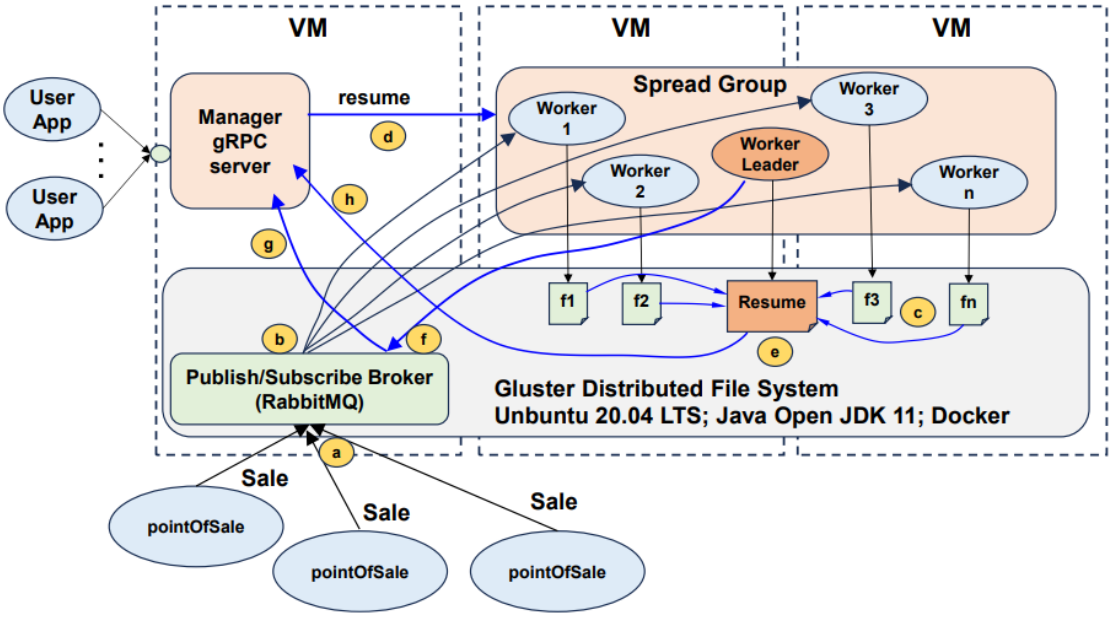
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA (ISEL)

DEPARTAMENTO DE ENGENHARIA ELETRÓNICA E DE TELECOMUNICAÇÕES E COMPUTADORES (DEETC)

MESTRADO EM ENGENHARIA DE INFORMÁTICA E MULTIMÉDIA (MEIM)
UNIDADE CURRICULAR DE COMPUTAÇÃO NA NÚVEM

Trabalho Final

Grupo GN03



Docente

Professor	Luís Assunção

Ana Sofia Oliveira (A39275)

Dezembro, 2023

Índice

Introdução

O presente relatório documenta o desenvolvimento de um sistema distribuído abrangente, projetado para otimizar a gestão de vendas num ambiente distribuído. Este sistema consiste no desenvolvimento de quatro aplicações: PointOfSales, UserApp, Server e Worker. Cada aplicação desempenha um papel fundamental na captura, processamento e análise de dados de vendas.

- **PointOfSales:** Aplicação desenvolvida para enviar transações de vendas para serem registadas no sistema. Recorre ao middleware RabbitMQ para enviar mensagens para um exchange conhecido, proporcionando uma comunicação eficiente e assíncrona entre as diferentes partes do sistema.
- **UserApp:** Aplicação de gestão das vendas. Permite que utilizadores solicitem a composição de resumos de vendas de uma determinada categoria (casa ou alimentar). Para além disso, permite ainda descarregar os respetivos relatórios de vendas. A comunicação com o servidor é facilitada através do contrato gRPC, garantindo uma comunicação rápida e eficaz.
- **Server:** O servidor é responsável por receber pedidos de resumo de vendas provinientes da aplicação UserApp e enviar o documento criado, implementando o contrato gRPC. Ao receber um pedido de resumo, o cliente encaminha a execução do pedido para um Grupo Spread, onde um dos workers registados no grupo agregam todos os ficheiros de registo num único documento. Estes pedidos ocorrem através de comunicação por eventos, pelo que, de forma que o servidor consiga ter conhecimento do término da tarefa, é criada uma estrutura por pedido realizado. Aquando do pedido de resumo do servidor para o Spread Group é igualmente enviada a informação do local para onde o worker líder irá responder. Para o download, o servidor envia o ficheiro disponível no sistema de ficheiros partilhados.
- **Worker:** Aplicação dedicada ao registo das vendas enviadas pela aplicação PointOfSales. Esta aplicação regista vendas de uma determinada categoria. Para isso, consome mensagens provenientes de uma fila dedicada à categoria pretendida (casa ou alimentar). Para além disso, a aplicação também agrega a informação de várias vendas compondo um relatório de vendas por categoria. Os pedidos de resumo são enviados para um Spread Group previamente definido e onde cada worker participa. Quando recebe uma mensagem para resumir as vendas, apenas um worker é eleito para realizar esta tarefa. Esta eleição é realizada através de um algoritmo de eleição simples, Bully Election. O documento criado ficará armazenado num sistema de ficheiros partilhado, pelo que tanto os restantes Workers, tal como o servidor terão acesso aos mesmos.

O desenvolvimento deste sistema distribuído foi realizado em Java, utilizando os middlewares RabbitMQ, GlusterFS, Spread e gRPC, que desempenham papéis essenciais na criação de uma arquitetura robusta, eficiente e escalável. Este relatório pretende fornecer uma visão abrangente da arquitetura, estrutura de dados e implementação do projeto, demonstrando o sucesso na criação de uma solução tecnologicamente avançada para a gestão de vendas distribuídas.

Desenvolvimento

Para a realização do sistema proposto foi necessário definir a arquitetura do sistema, bem como a arquitetura de cada estrutura de dados utilizada. A arquitetura geral encontra-se abaixo.

Registo de vendas

O registo de vendas é iniciado pela aplicação PointOfSales, que envia uma mensagem com o produto vendido para um exchange no RabbitMQ previamente conhecido. Este exchange distribui as mensagens de acordo com a sua exchange key entre as filas das categorias alimentar ou casa. Posteriormente, a aplicação worker, consome as mensagens da fila que corresponde à categoria dos produtos que irá registar.

Mensagens

Cada venda é identificada e enviada para registo com as seguintes informações:

- Data da venda;
- Nome do produto;
- Categoria do produto;
- Quantidade vendida;
- Preço (com e sem IVA).

Assim, foi definida a mensagem SalesMessage cujo formato se encontra exemplificado abaixo.

```
{
  "produto":"bide",
  "data":"dez 13,2023 20:23:19",
  "categoria":"casa",
  "quantidade":1,
  "precoSIVA":54.99,
  "precoCIVA":67.6377
}
```

Estrutura RabbitMQ

De forma a garantir a funcionalidade acima descrita, foi definido o exchange “ExgSales” do tipo topic. Este exchange tem duas filas associadas “alimentar” e “casa” para onde as mensagens são encaminhadas de acordo com as routing keys “alimentar.#” e “casa.#”, respetivamente.

Assim, foi definida a estrutura apresentada abaixo, onde as mensagens a vermelho representam vendas com a categoria “casa” e as mensagens a azul representam vendas com a categoria “alimentar”.

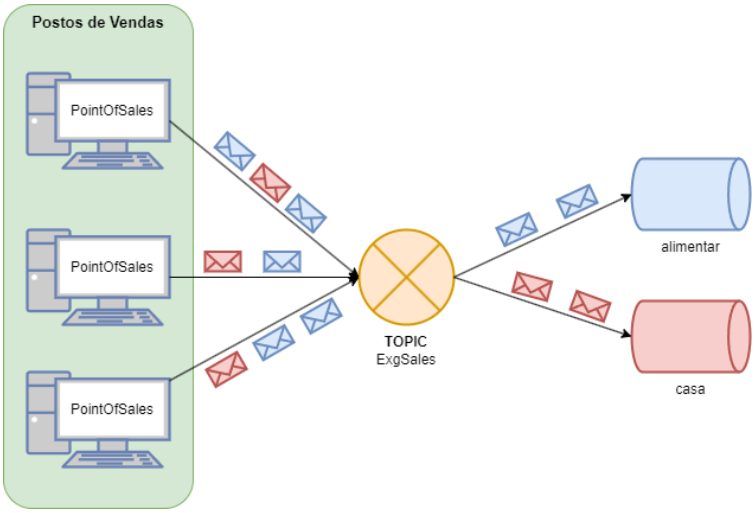


Figura 1 - Estrutura RabbitMQ para registo de vendas

Pedido de resumo

O pedido de resumo inicia-se na aplicação UserApp, que envia um pedido para o servidor. Por sua vez, o servidor envia uma mensagem para o Spread Group onde os eorkers pertencem. Ao receber o pedido do servidor, os workers elegem um líder entre si para realizar a tarefa. O líder agrupa os ficheiros de vendas e, após isso, responde ao servidor indicando o término da tarefa. Após isto, o servidor encaminha para o cliente o identificador único do documento gerado.

Para a realização desta tarefa, foi necessário definir diversos componentes no sistema. Irei detalhar cada componente de seguida.

Contrato gRPC

O contrato gRPC define as operações entre a aplicação UserApp e o servidor. Tal como indicado anteriormente, o cliente faz pedidos para resumir as vendas e download dos documentos.

Este contrato define as operações agruparVendas e obterFicheiro. Para a operação agruparVendas, o cliente deve identificar a categoria e o servidor irá responder a este pedido com informações sobre o documento, nomeadamente o identificador único do mesmo.

Através do identificador único do documento criado, o cliente poderá utilizar a operação obterFicheiro. Na resposta o cliente enviará ao cliente o documento por blocos, identificando o identificador único, o nome do ficheiro e blocos do documento.

```
service GestService {
    rpc agruparVendas(Categoria)  returns (Informacoes);
    rpc obterFicheiro(Informacoes)  returns (stream Ficheiro);
}

message Categoria {
    string tipo = 1;
}

message Informacoes {
    string hash = 1;
}

message Ficheiro {
    string hash = 1;
    string nomeFicheiro = 2;
    bytes blocos = 3;
}
```

Estrutura RabbitMQ

Quando recebe um pedido para criação resumo das vendas, o servidor envia uma mensagem para o Spread Group onde os Workers estão registados (ver estrutura spread group). Após a conclusão da tarefa de resumo, o worker líder envia uma notificação ao servidor. Para que exista notificação seja realizada por pedido e tirando partido do facto de que cada pedido de resumo gRPC tem o seu fio de execução, foi criada uma estrutura própria para resposta aos pedidos de resumo. Antes de realizar o pedido ao Spread Group, o servidor cria uma estrutura dedicada para resposta ao pedido através do RabbitMQ. Esta estrutura é composta por:

- Um exchange do tipo fanout com o nome composto entre a expressão "ex_" e o id do pedido;
- Queue com o nome composto entre a expressão "queue_" e o id do pedido.

Após consumir a mensagem, o servidor deverá elimina esta estrutura de forma a minimizar o número de recursos utilizados.

A estrutura detalhada acima encontra-se representada na figura em baixo.

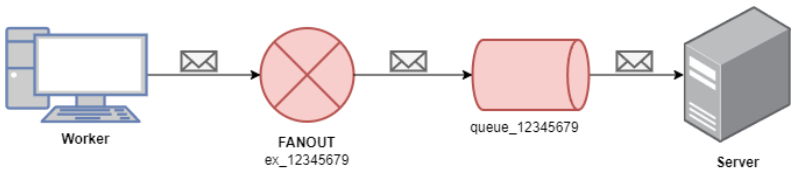


Figura 2 – Estrutura Rabbit MQ por pedido resumo

Estrutura Spread Group

Todos os workers pertencem a um Spread Group com nome “workers”. As mensagens são enviadas por multicast pelo que todos os elementos do grupo recebem as mensagens enviadas.

Neste processo podem existir dois tipos de mensagens: externas ou internas ao grupo. De forma a simplificar o processo, todas as mensagens enviadas têm o seguinte formato JSON:

```
{
  "id":123456,
  "value":"START_ELECTION",
  "exchange":"ex_123456",
  "tipo":"casa"
}
```

Quando é enviada uma mensagem externa, todos os atributos da mensagem deverão ter informação. Isto porque, é através das informações nesta mensagem que o worker líder irá saber para onde deve encaminhar a resposta.

Quando é envida uma mensagem interna, apenas os atributos id e value é deverão estar preenchidos. De notar que, em mensagens internas o atributo value apenas assume os seguintes valores: RESUME, START_ELECTION, OK ou COORDINATOR, de forma a facilitar o processo de eleição do worker líder.

Processo de eleição

A mensagem enviada pelo servidor ao Spread Group é encaminhada com o valor START_ELECTION de forma a iniciar de imediato uma eleição dentro do grupo. Cada worker pode ter três estados possíveis:

- WORKING: A realizar a tarefa de registo das vendas;
- IN_ELECTION: Potencial candidato para ganhar a eleição;
- LIDER: A realizar a tarefa de resumo das vendas.

Ao receber mensagens o worker irá atualizar o seu estado de acordo com a informação recebida.

O algoritmo de eleição implementado é o Bully Election, onde se mantém em processo de eleição apenas os workers que ainda estão em processo de eleição. Este algoritmo funciona da seguinte forma:

1. Worker recebe mensagem RESUME, altera do seu estado para IN_ELECTION e enviada nova mensagem START_ELECTION para iniciar uma eleição;
2. Worker recebe mensagem START_ELECTION e opera mediante a comparação do seu identificador com o identificador do remetente da mensagem:
    a. ID Worker > ID Remetente: Envia mensagem OK, volta a enviar mensagem START_ELECTION e espera que lhe seja enviada mensagem OK;
    b. ID Worker < ID Remetente: Altera o seu estado para WORKING.

3. Worker não recebe mensagem de resposta OK após ter enviado mensagem START_ELECTION: Altera o seu estado para LIDER e envia mensagem COORDINATOR indicando que é o novo líder.
4. Worker recebe mensagem COORDINATOR: Altera o seu estado para WORKING, aceitando a eleição.

Sempre que é enviada uma mensagem de eleição (START_ELECTION), o worker inicia uma contabilização do tempo máximo de duração da eleição (1 segundo). Se a eleição não terminar dentro desse tempo e o worker se mantiver como potencial candidato à vitória, envia uma mensagem a autoassumir-se líder.

O diagrama de estados do Worker é apresentado na imagem abaixo.

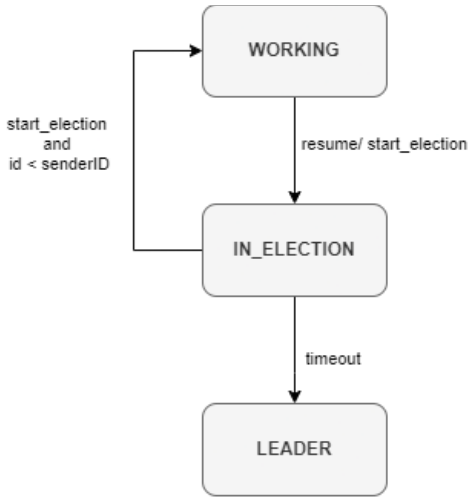


Figura 3 - Diagrama de estados Worker

Este processo é ilustrado na imagem abaixo, onde os Workers 1, 10 e 5 despoletam um processo de eleição. Podemos verificar que inicialmente todos os Workers recebem emitem e recebem mensagem START_ELECTION. Apenas respondem aqueles cujo ID é superior ao ID do remetente, com mensagem OK, e voltam a enviar nova mensagem START_ELECTION recebida por todos os elementos do grupo. Este processo repete-se até que deixem de existir respostas a mensagens START_ELECTION e seja enviada mensagem COORDINATOR, em consequência do tempo de espera.

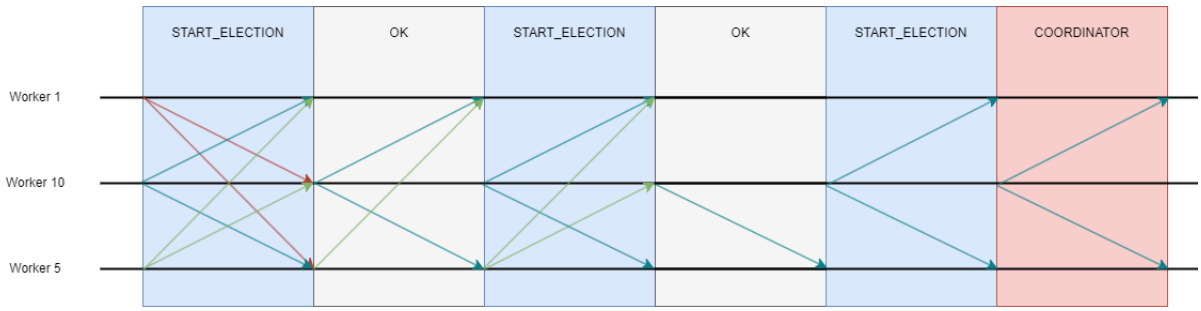


Figura 4 - Algoritmo de eleição

Conclusões

O desenvolvimento deste sistema distribuído proporciona uma solução eficaz e escalável. A arquitetura do sistema, composta por quatro aplicações distintas - PointOfSales, UserApp, Server e Worker - foi cuidadosamente projetada para atender às necessidades específicas da captura, processamento e análise de dados de vendas.

A aplicação PointOfSales utiliza o middleware RabbitMQ para enviar transações de vendas de forma assíncrona e eficiente para serem registadas no sistema. A UserApp, por meio de um contrato gRPC, permite aos usuários solicitar resumos de vendas e realizar o download dos respetivos relatórios, garantindo uma comunicação rápida e eficaz com o servidor.

O servidor desempenha um papel central, recebendo pedidos de resumo da UserApp, encaminhando a execução para um grupo Spread de workers, e gerenciando a conclusão da tarefa. O algoritmo de eleição Bully Election é implementado para selecionar um líder entre os workers, otimizando a eficiência do processo de resumo de vendas.

A aplicação Worker, responsável pelo registo de vendas, consome mensagens categorizadas provenientes do PointOfSales, agregando informações para compor relatórios de vendas por categoria. O uso de tecnologias como RabbitMQ, GlusterFS, Spread e gRPC contribui para a criação de uma arquitetura robusta e eficiente.

Em resumo, o sistema desenvolvido em Java destaca-se pela sua capacidade de lidar com a complexidade inerente à gestão de vendas em um ambiente distribuído, oferecendo uma solução que se destaca pela eficiência na comunicação entre os componentes e na manipulação de grandes volumes de dados de vendas. Este relatório fornece uma visão abrangente da arquitetura, estrutura de dados e implementação do projeto, demonstrando o sucesso na criação de uma solução tecnologicamente avançada para a gestão de vendas distribuídas.