	<p>Departamento de Engenharia de Eletrónica de Telecomunicações e Computadores</p> <p>Mestrado em Engenharia Informática e de Computadores (MEIC)</p> <p>Mestrado em Engenharia Informática e Multimédia (MEIM)</p>
<p>17-11-2023</p>	<p>Computação Distribuída (Inverno 23/24)</p>

## Trabalho Prático de Avaliação Final

**Objetivo:** Implementação de um sistema distribuído englobando os diferentes paradigmas de interação e *middleware* estudados e já utilizados nos Laboratórios das aulas práticas

### Notas prévias:

- Embora possam já existir períodos de dúvidas nas aulas da semana de 20/Nov, as aulas das semanas de 27/Nov, 04/Dez e parcialmente de 12/Dez de 2023, serão totalmente alocadas para apoio à realização do trabalho. No entanto, é pressuposto, e faz parte dos ECTS da Unidade Curricular, que cada grupo de alunos terá de dedicar horas de trabalho fora das aulas. Para eventual apoio e esclarecimento de dúvidas fora das aulas devem agendar com os professores o pedido de ajuda que poderá ser feito presencial ou remoto via Zoom. (nos *links* disponíveis no Moodle de cada turma);
- De acordo com as regras de avaliação definidas no slide 5 do conjunto *CD-01 Apresentação.pdf*, este trabalho tem um peso de 30% na avaliação final e é de entrega obrigatória, com avaliação de nota mínima de 10 valores;
- A entrega será realizada em Moodle com um ficheiro Zip, incluindo os projetos desenvolvidos (*src* e *pom.xml* sem incluir os artefactos JAR), bem como outros ficheiros que considerem pertinentes para valorizar a avaliação do trabalho. É obrigatório a entrega de documento PDF como um relatório técnico que descreve o sistema implementado, permitindo a um leitor compreender o objetivo, pressupostos, a arquitetura, a configuração para execução do sistema e as conclusões. A qualidade deste relatório terá peso significativo na avaliação final do trabalho;
- Na penúltima e última semana do semestre (14 a 20 de dezembro de 2023) cada grupo terá de apresentar e demonstrar, durante 15 minutos e para toda a turma, a funcionalidade e operacionalidade do trabalho realizado, sendo a calendarização em cada turma comunicada posteriormente;
- A entrega limite no Moodle será 13 de dezembro de 2023 até às 23:59h.

Foi decidido implementar uma plataforma informática de uma cadeia de supermercados com vários componentes que se executam em múltiplas máquinas virtuais (VM) na *Google Cloud Platform* (Figura 1), configuradas com um sistema de ficheiros distribuídos (*Gluster file System*) que permite que qualquer ficheiro criado numa VM seja replicado em todas as VM.

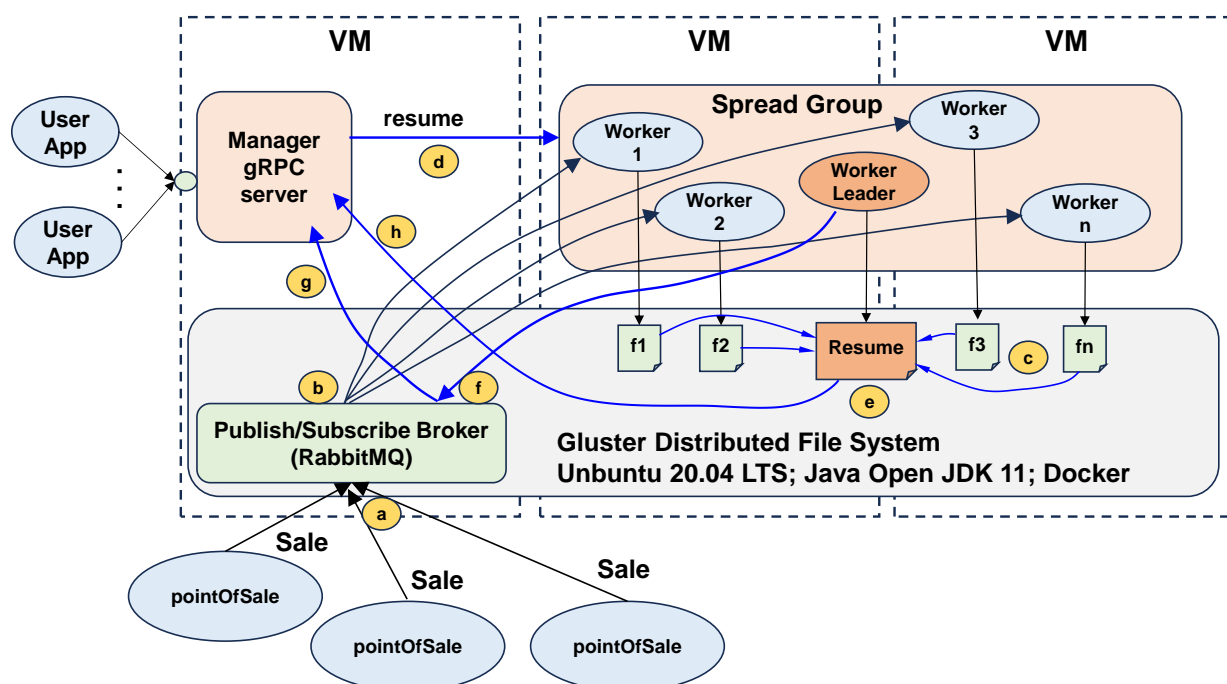



Figura 1 - Diagrama geral do sistema

 <b>ISEL</b> <small>INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA</small>	<b>Departamento de Engenharia de Eletrónica de Telecomunicações e Computadores</b> <b>Mestrado em Engenharia Informática e de Computadores (MEIC)</b> <b>Mestrado em Engenharia Informática e Multimédia (MEIM)</b>
<b>17-11-2023</b>	<b>Computação Distribuída (Inverno 23/24)</b>

A cadeia de supermercados tem múltiplos pontos de vendas (*PointOfSale*) que por cada produto vendido geram uma mensagem **(a)**, sendo que o conteúdo de uma venda (*Sale*) tem as seguintes informações (*Data*, *CodigoProduto*, *NomeProduto*, *Quant*, *PreçoUnitário*, *Total*, *Iva*).


Pretende-se desenvolver um sistema distribuído com as componentes ilustradas na Figura 1 e requisitos a seguir descritos.

### Requisitos funcionais

- Desenvolver uma pequena aplicação (*PointOfSale*) que simula uma caixa de supermercado e que envia **(a)** mensagens (*Sale*) de vendas para um *Broker Publish/Subscriber* implementado como um Docker container, onde existe um *exchange* global bem conhecido de nome *ExgSales*;
- Os produtos têm unicamente duas categorias: ALIMENTAR ou CASA, pelo que cada mensagem *Sale* é enviada com os valores possíveis na *routing Key* ALIMENTAR.# ou CASA.#, em que # representa quaisquer outras palavras relacionadas com a venda (*Sale*);
- Usando o padrão *work-queue* existem múltiplas instâncias de uma aplicação *Worker* que subscrevem filas no Broker **(b)**, para processarem as mensagens de vendas. Cada *Worker* só trata de um tipo de vendas (ALIMENTAR ou CASA);
- Os *Workers* vão escrevendo **(c)** em ficheiros (f1, f2, ...,fn) as mensagens (*Sale*), em que cada venda é uma linha de texto num ficheiro. Os ficheiros devem ser criados em diretoria que estejam partilhadas no sistema de ficheiros distribuídos *Gluster*, pelo que todos esses ficheiros serão replicados e acessíveis em todos os nós computacionais (VM);
- Todos os *Workers* pertencem a um grupo Spread, pelo que mensagens enviadas para o grupo em *multicast* serão recebidas por todos os membros do grupo, isto é, por todos os *Workers*;
- Periodicamente através da aplicação *User App* utilizadores gestores de vendas, podem através do servidor Manager gRPC, emitir ordens de resumo de vendas com o objetivo de obter um único ficheiro com a junção de todas as vendas até esse momento (junção dos ficheiros f1, f2,...,fn). Para tal o servidor Manager gRPC, envia uma mensagem *resume* **(d)** em *multicast* para o grupo de *Workers* pedindo a operação de resumo de vendas de produtos ALIMENTAR ou CASA, indicando o nome de um *Exchange* para onde deve ser enviada a notificação **(f)** que o resumo está realizado, incluindo o nome do ficheiro onde deve ser escrito o resumo;
- Os *Workers* que processam as mensagens do tipo requerido para resumo, deixam de processar mais mensagens (*negative acknowledge*) e realizam uma eleição para eleger um *Worker Leader* que irá processar a junção dos ficheiros dos *Workers*, produzindo assim o ficheiro com o resumo de vendas **(e)**. O *Leader* notifica o fim da operação de resumo **(f)** enviando uma mensagem para o *Exchange* que foi indicado no pedido *resume* **(d)**, indicando o nome do ficheiro que contém o resumo de vendas;
- A mensagem de notificação será processada **(g)** pelo servidor Manager gRPC, possibilitando aos utilizadores da *User App*, fazerem download do ficheiro de resumo de vendas **(h)**;
- Logo que todos os *Workers* aceitam a eleição do *Leader*, devem reiniciar o processamento das mensagens de vendas passando escrevê-las num novo ficheiro. Da mesma forma o *Leader*, depois de notificar o fim do trabalho de resumo, passa novamente ao papel de *Worker*;
- O algoritmo de eleição deve ser o mais simples possível e que tire o máximo de partido da existência de mensagens *multicast* e de *Membership* nos grupos Spread. O algoritmo de eleição é um aspeto crucial na avaliação do trabalho pelo que a sua descrição e demonstração de funcionalidade deve ser cuidadosamente detalhada no relatório final;
- A definição do contrato do servidor Manager gRPC deve seguir princípios de simplicidade e funcionalidade adequada ao problema, sendo da total liberdade de cada grupo de alunos.

### Requisitos não funcionais

- Assuma que todo o trabalho vai unicamente necessitar de 3 nós computacionais (3 VM) pelo que pode considerar que todos os nós computacionais (VM) têm a mesma configuração;


 <b>ISEL</b> <small>INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA</small>	<b>Departamento de Engenharia de Eletrónica de Telecomunicações e Computadores</b> <b>Mestrado em Engenharia Informática e de Computadores (MEIC)</b> <b>Mestrado em Engenharia Informática e Multimédia (MEIM)</b>
<b>17-11-2023</b>	<b>Computação Distribuída (Inverno 23/24)</b>

- Utilize VM *instances* GCP do tipo *e2-medium* (2 vCPU, 1 core, 4 GB memory) com sistema operativo Ubuntu versão 20.04 LTS;
- Utilize o guia [install-configure-Vms.txt](#), ficheiro em anexo, onde pode encontrar instruções e os comandos para instalar/configurar as VMs com os *middleware* e *runtimes* necessários para o sistema operativo Ubuntu 20.04 LTS:
  - ✓ Compilador GCC e outras tools necessárias para compilar as sources do Spread Toolkit
  - ✓ Java Open JDK 11;
  - ✓ Docker runtime;
  - ✓ Instalação/configuração do Gluster File System (mais informação em <https://www.gluster.org/>);
  - ✓ Instalação/configuração do Spread Toolkit (mais informação em <http://www.spread.org/>).
- A atribuição às 3 VM dos vários componentes (RabbitMQ, Manager Server e Workers) do sistema a desenvolver, é definida por cada grupo seguindo uma estratégia que considere oportuna;
- Tanto na plataforma RabbitMQ como no Spread os dados das mensagens são normalmente um *array* de bytes (`byte[]`). No entanto, para maior flexibilidade deve ser possível, nas várias aplicações a desenvolver, usar classes Java para definir as mensagens a transferir entre os diversos intervenientes. Por exemplo, as mensagens entre o grupo de *workers* deve ser feita com mensagens Spread que transportam a serialização de objetos no formato JSON. Sugere-se a utilização da biblioteca Gson, <https://mvnrepository.com/artifact/com.google.code.gson/gson/> disponível no repositório central Maven, que de forma simples e flexível permite fazer conversões de objetos para `byte[]` e de `byte[]` para objetos. Em anexo, no final do enunciado, apresenta-se um exemplo de uso da referida biblioteca (Gson);

### Sugestões Gerais

- Qualquer questão ou dúvida sobre os requisitos deve ser discutida com o professor;
- Antes de começar a escrever código desenhe a arquitetura do sistema, os contratos de interação bem como os diagramas de interação mais importantes, nomeadamente as trocas de mensagens necessárias ao algoritmo de eleição;
- Quando tiver dúvidas sobre os requisitos, verifique no site *Moodle* se existem “*Frequently Asked Questions*” com esclarecimentos sobre o trabalho;
- Parametrize todas as aplicações por forma a ser possível fazer *deployment* do sistema em múltiplas VM, sem portos e endereços TCP/IP *hard-coded*. Sugere-se a utilização nas várias aplicações de argumentos na linha de comando, por exemplo:
 

```
java -jar worker.jar <ipRabbitMQ> <portRabbitMQ> <workQueue> <...>
```
- Inscreva no código ou em ficheiros *readme.txt* descrições sucintas e justificativas das partes mais relevantes;
- Não esqueça o relatório é parte importante e terá peso na avaliação final do trabalho.

 <b>ISEL</b> <small>INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA</small>	Departamento de Engenharia de Eletrónica de Telecomunicações e Computadores Mestrado em Engenharia Informática e de Computadores (MEIC) Mestrado em Engenharia Informática e Multimédia (MEIM)
17-11-2023	Computação Distribuída (Inverno 23/24)

## Anexo: Exemplo de conversão: Objeto -> byte[] -> Objeto

```

public static void main(String[] args) {
    SomeClass someObject=new SomeClass();
    someObject.setId(5); someObject.setName("ABCD");
    someObject.setResults(new String[]{"abc","def"});
    System.out.println(someObject.toString());
    // converter objeto em string JSON
    Gson js=new GsonBuilder().create();
    String jsonString=js.toJson(someObject);
    System.out.println(jsonString);
    // converter string em byte[]
    byte[] binData=jsonString.getBytes(StandardCharsets.UTF_8);
    for (byte b : binData) System.out.print(b+" ");
    System.out.println();
    // binData pode ser enviado como mensagem em binário
    // em qualquer plataforma por exemplo RabbitMQ ou Spread,...
    // Receção de binData e deserialização para objeto
    String newJsonString=new String(binData, StandardCharsets.UTF_8);
    SomeClass newSomeObject=js.fromJson(newJsonString,SomeClass.class);
    System.out.println(newSomeObject.toString());
}

```

```

public class SomeClass {
    private int id;
    private String name;
    private String[] results;

    public SomeClass(){}

    public int getId() { return id; }
    public void setId(int id) { this.id = id; }
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public String[] getResults() { return results; }
    public void setResults(String[] results) { this.results = results; }

    @Override
    public String toString() {
        String strResults="["; boolean first=true;
        for (String s : getResults()) {
            strResults+= first? "\"" + s + "\"" : "," + "\"" + s + "\""; first=false;
        }
        strResults+="]";
        return "SomeClass("+getId()+","+getName()+","+strResults+")";
    }
}

```

```

<dependency>
<groupId>com.google.code.gson</groupId>
<artifactId>gson</artifactId>
<version>2.10.1</version>
</dependency>

```