Trabalho Prático de Avaliação Final (TPA2)

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 24/Nov, as aulas das semanas de 01/Dez e 08/Dez de 2025**, serão essencialmente 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 Teams ou 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 9,5 valores;
- A entrega será realizada em Moodle com um ficheiro Zip, incluindo os projetos desenvolvidos (*src,* pom.xml e *assembly.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 última semana do semestre (15 a 20 de dezembro de 2025)** cada grupo terá de apresentar e demonstrar, durante 10 minutos, para toda a turma, a funcionalidade e operacionalidade do trabalho realizado, sendo a calendarização em cada turma comunicada posteriormente;
- **A entrega limite no Moodle é 13 de dezembro de 2025 até às 23:59h**.

Considere um cenário onde existe um repositório distribuído com ficheiros cujo conteúdo são mensagens de email trocadas entre múltiplas pessoas, nomeadamente numa equipa de desenvolvimento de sistemas distribuídos (ver exemplo de um email no **Anexo 1**).
Pretende-se desenvolver um sistema que permita a múltiplos utilizadores submeter pedidos de pesquisa no repositório de mensagens de email que contenham um determinado conjunto de *substrings*.
Na Figura 1 apresenta-se um diagrama geral do sistema com o objetivo de identificar as partes envolvidas, bem como as principais interações entre elas. Note-se que este diagrama é meramente uma vista geral e não um desenho da arquitetura do sistema pois esta deverá identificar mais claramente as componentes de software envolvidas e que deverá ser apresentada e descrita no relatório final do trabalho.

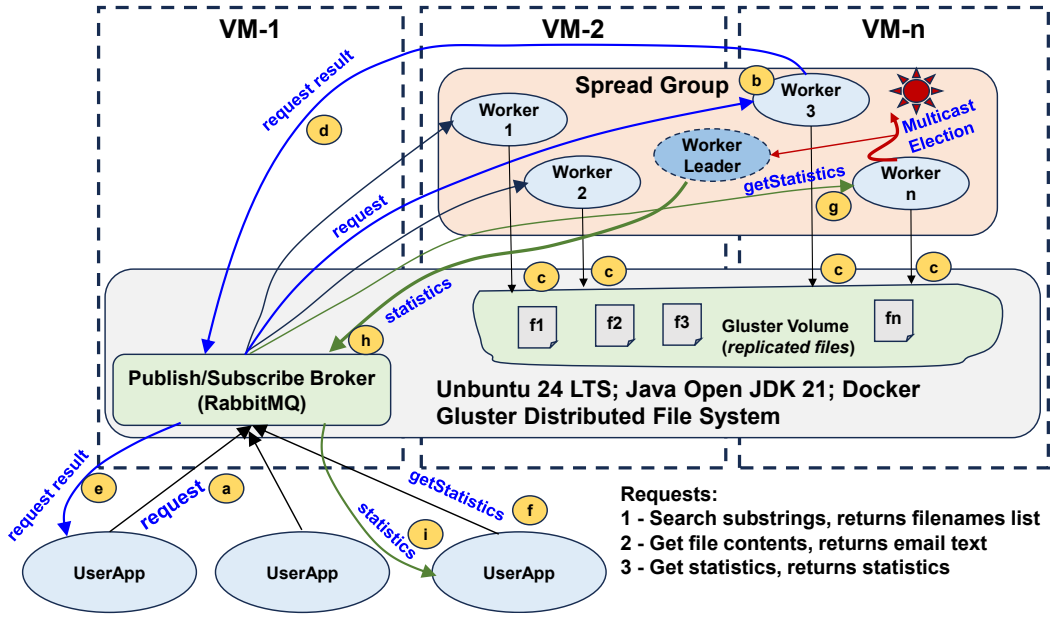

Figura 1 – Diagrama geral do sistema

O sistema a desenvolver deve permitir que utilizadores de uma aplicação (*UserApp*) possam:

- Obter uma lista com os nomes dos ficheiros que contêm em simultâneo um determinado conjunto de *substrings*, por exemplo {"gRPC em Java 21", "GCP", "Docker"};
- Dado o nome de um ficheiro se possa obter o conteúdo desse ficheiro;
- Obter estatísticas sobre o comportamento do sistema, nomeadamente qual o número total de pedidos efetuados globalmente no sistema, incluindo os que obtiveram pelo menos um resultado mais o número de pedidos que não obtiveram qualquer resultado.

Requisitos funcionais

- O sistema baseia-se no modelo *publish-subscribe* garantindo que os múltiplos pedidos, realizados pelos utilizadores, através da aplicação *UserApp*, consistem no envio (*publish*) de mensagens para um *Broker Publish-Subscribe*. As respostas aos pedidos específicos de cada utilizador são obtidas assincronamente (*subscribe*) de filas (*queues*) existentes num *Broker Publish-Subscribe*;
- Existem 3 tipos de pedidos (mensagens publicadas para o *Broker Publish-Subscribe*):
    - Pedido de lista de nomes de ficheiros que contêm em simultâneo um conjunto de *substrings*;
    - Pedido para obter o conteúdo integral (*string*) de um determinado ficheiro;
    - Pedido de obtenção de estatísticas.
- As respostas aos pedidos são obtidas assincronamente como mensagens *subscribed* a partir do *Broker Publish-Subscribe*:
    - Lista de nomes de ficheiros que contêm o conjunto de *substrings*;
    - Conteúdo (*string*) de um determinado ficheiro;
    - Estatísticas (número global de pedidos e número dos que tiveram sucesso).
- Para suportar distribuição de carga, os pedidos são processados por várias instâncias de uma aplicação *Worker* (padrão *work-queue*), que podem executar-se em múltiplas máquinas virtuais;
- As várias instâncias *Worker* são membros de um grupo com comunicação *multicast* e suporte de *membership* entre os membros do grupo;
- Como *se* indica na Figura 1, uma *UserApp* envia um pedido como uma mensagem de *request* (a) para o *Broker*. De acordo com o *work-queue*, o *Worker* 3 recebe essa mensagem (b) e de acordo com o conjunto de *substrings*, indicadas no pedido, pesquisa em todo o repositório (c) pelos ficheiros que contêm essas *substrings*. A resposta ao pedido (d) deve ser enviada para o *Broker* de acordo com as indicações passadas no pedido. Finalmente, a *UserApp* recebe (e) do *Broker* a resposta;
- Os ficheiros com as mensagens de email (ficheiros com extensão .txt) existem numa única diretoria e estão replicados em múltiplas máquinas virtuais usando um *File System* distribuído. Anexo a este enunciado é disponibilizado o ficheiro ***EmailFiles.zip*** que contém 20 ficheiros de texto em que cada um tem uma mensagem de email. Este conjunto de ficheiros deve ser replicado e partilhado, pelo menos em todas as VM onde se executam *Workers*;
- No **Anexo 2**, apresenta-se uma aplicação Java que ilustra como cada *Worker* poderá percorrer uma diretoria de ficheiros .txt e obter os nomes dos ficheiros e respetivo conteúdo que contêm um conjunto de *substrings*;
- Quando a *UserApp* envia um pedido (f) de obtenção de estatísticas (*getStatistics*) o *worker* que recebe (g) o pedido (na Figura 1 o *Worker n*) comunica *multicast* com todo o grupo de *workers* para que se atinga o consenso de qual dos *workers* vai coordenar a obtenção das estatísticas parciais de todos os *workers*, sumarizando os totais do número de pedidos satisfeitos e não satisfeitos. Por consenso, será o *worker* coordenador que responde ao pedido (h). Finalmente a *UserApp* recebe (i) do *Broker* a resposta com os valores das estatísticas;

Requisitos não funcionais

- O sistema executa-se em VM da plataforma Google Cloud Plataform (GCP);
- As VM no GCP devem ter o sistema operativo Ubuntu 24 LTS, Java 21 e Docker runtime;
- Todas as VM partilham o repositório dos ficheiros de email através de um volume do File System distribuído Gluster com replicação de ficheiros;
- O broker *publish-subscribe* deve ser o RabbitMQ em execução como container Docker numa VM da plataforma GCP;
- As aplicações envolvidas (*UserApp* e *Worker*) devem ser desenvolvidas em Java 21;
- A definição das mensagens de pedido e de resposta devem seguir princípios de simplicidade e funcionalidade adequada ao problema, sendo da total liberdade de cada grupo de alunos;
- A estrutura de dados interna do *Worker* para manutenção de estado do número de pedidos com sucesso e não sucesso é da inteira responsabilidade de cada grupo de alunos;
- A comunicação por grupo em *multicast* é suportada pelo Spread Toolkit instalado nas VM do GCP;
- Assuma que a demonstração da funcionalidade final do sistema vai unicamente necessitar de 3 nós computacionais (3 VM) pelo que pode considerar que todos os nós computacionais (VM) têm a mesma configuração;
- Utilize VM *instances* GCP com sistema operativo Ubuntu versão 24 LTS, tal como fez durante os Laboratórios e no trabalho prático TPA1;
- Utilize os guias disponibilizados ao longo dos laboratórios com as instruções e os comandos para instalar/configurar as VMs com os *middleware* e *runtimes* necessários para o sistema operativo Ubuntu 24 LTS:
  - ✓ Java Open JDK 21;
  - ✓ Docker runtime;
  - ✓ Compilador GCC e outras *tools* necessárias para compilar as *sources* do Spread *toolkit*;
  - ✓ Instalação/configuração do **Spread Toolkit de acordo com a realização do Laboratório 04**;
  - ✓ Para instalação/configuração do Gluster File System (mais informação em https://www.gluster.org/), siga as instruções no **Anexo 3**;
  - ✓ O volume Gluster deve ter inicialemente os 20 ficheiros de texto com emails fornecidos no anexo ao enunciado ***EmailFiles.zip***. No entanto, a demonstração da funcionalidade do sistema deve permitir acrescentar novos ficheiros numa das VM que serão replicados nas outras VM;
- A atribuição às 3 VM dos vários componentes (*RabbitMQ, e Workers*) do sistema a desenvolver, é definida por cada grupo seguindo uma estratégia que considere adequada;
- Tanto na plataforma *RabbitMQ* como no Spread *toolkit* os dados das mensagens são normalmente um *array* de bytes (`byte[]`). No entanto, para maior flexibilidade, devem nas várias aplicações usar classes Java para definir as mensagens a transferir entre os diversos intervenientes. Sugere-se a utilização da biblioteca Gson para a serialização de objetos no formato JSON, disponível no repositório central Maven (https://mvnrepository.com/artifact/com.google.code.gson/gson/), que de forma simples e flexível permite fazer conversões de objetos para `byte[]` e de `byte[]` para objetos. No **Anexo 4**, apresenta-se um exemplo de uso da referida biblioteca (Gson);
- O algoritmo de Consenso para eleger o coordenador de obtenção de estatísticas deve ser o mais simples possível e que tire o máximo partido da existência de mensagens *multicast* e de *Membership* no Spread. O algoritmo de eleição é um aspeto importante na avaliação do trabalho pelo que a sua descrição e demonstração de funcionalidade deve ser cuidadosamente detalhada no relatório final;

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 e os diagramas de interação mais importantes, nomeadamente as mensagens necessárias ao algoritmo de consenso;
- 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> <Spread Group> <...>
```

- Inscreva no código comentários ou em ficheiros *readme.txt* descrições sucintas e justificativas das partes mais relevantes;
- Não esqueça que o relatório é parte importante e terá peso na avaliação final do trabalho.

NOTA IMPORTANTE:

Quaisquer ideias de implementar o sistema com requisitos funcionais e não funcionais ou outras tecnologias não constantes neste enunciado, deve ser previamente discutida e validada pelo professor. Em caso contrário, o trabalho entregue terá avaliação nula.

Anexo 1: ficheiro email017.txt

Como exemplo, este email deve ser encontrado perante a pesquisa das seguintes *substrings*: **"gRPC em Java 21", "GCP" e "Docker".**

```
De: rodrigo.santiago@techteam.pt
Para: manuela.afonso@techteam.pt, antonio.silva@techteam.pt
Assunto: Protótipo gRPC em Java concluído
Data: 2025-11-12

Manuela e António,

Boas notícias! Terminei o protótipo de serviço gRPC em Java 21.
Implementação:
- Definição do serviço em Protocol Buffers (.proto)
- Server gRPC com múltiplos métodos (unary, streaming)
- Cliente Java para testes
- Exemplos de uso incluídos
Resultados dos testes:
- Latência média: 5ms (vs 45ms com REST)
- Throughput: 10.000 req/s (vs 2.000 com REST)
- Uso de CPU: 30% inferior ao REST
- Tamanho das mensagens: 60% menor com protobuf
O código está no repositório: github.com/techteam/grpc-prototype
Próximos passos:
1. Integrar com RabbitMQ para mensageria assíncrona
2. Deploy em containers Docker
3. Testes nas VMs da plataforma GCP
4. Implementar autenticação e encriptação

Manuela, podes fazer code review?

Abraço,
Rodrigo
```

Anexo 2: Aplicação de processamento de ficheiros .txt num diretoria

A aplicação percorre todos os ficheiros .txt existentes numa diretoria e mostra no ecrã o nome do ficheiro e o respetivo conteúdo que contém em simultâneo um conjunto de *substrings* passado como argumento na linha de comandos.

```
public static void main(String[] args) {
   // args[0] directory path args[1]...args[n] substrings to search
   String directoryPath = null;    List<String> substringsList = new ArrayList<>();
   try {
      if (args.length > 0) {
          directoryPath = args[0];
          for (int i = 1; i < args.length; i++)   substringsList.add(args[i]);
      } else {   System.out.println("Nothing to search!");    exit(-1);   }
      // Map<finename, emailText>
      Map<String, String> matchingEmails = searchInsideEmails(directoryPath, substringsList);
      System.out.println(" There is " + matchingEmails.size() + " email(s) containing all substrings\n");
      for (String fileName : matchingEmails.keySet()) {
         System.out.println("##:" + file);
         System.out.println(matchingEmails.get(fileName));
      }
   } catch (IOException e) {
      System.err.println("Error when processing  " + directoryPath + " files \n" + e.getMessage());
   }
}

public static Map<String, String> searchInsideEmails(String directoryPath, List<String> substringsList)
      throws IOException {
   Map<String, String> matchingEmails = new HashMap<>();
   // Percorrer todos os ficheiros .txt na diretoria
   Files.list(Paths.get(directoryPath))
         .filter(Files::isRegularFile)
         .filter(path -> path.toString().endsWith(".txt"))
         .forEach(path -> {
            try {
               String emailMessage = Files.readString(path);
               if (containsAllSubstrings(emailMessage, substringsList)) {
                  matchingEmails.put(path.toString(), emailMessage);
               }
            } catch (IOException e) {
               System.err.println("Read error in file: " + path + " - " + e.getMessage());
            }
         });
   return matchingEmails;
}

public static boolean containsAllSubstrings(String message, List<String> substringsList) {
   String lowerMessage = message.toLowerCase();
   for (String substr : substringsList) {
      if (!lowerMessage.contains(substr.toLowerCase())) {   return false;    }
   }
   return true;
}
```

Anexo 3: Instalação do Sistema de ficheiros distribuídos Gluster

Numa VM base com sistema operativo Ubuntu 24 LTS, para além da instalação dos ambientes já usados nos laboratórios (Java, Docker e Spread Toolkit) deve agora instalar o sistema de ficheiros distribuídos Gluster (https://www.gluster.org/).

```
# Instalação do Gluster
sudo add-apt-repository ppa:gluster/glusterfs-11
sudo apt update
sudo apt install glusterfs-server
# Lançar o daemon do Gluster e verificar que está em execução
sudo service glusterd start
# Observar o estado do serviço gluster
sudo service glusterd status
# Criar diretoria com permissões totais para ser um Brick de um gluster volume
sudo mkdir -p /var/gluster/brick
sudo chmod 777 /var/gluster/brick
# Criar diretoria para armazenar ficheiros das aplicações que criam/usam ficheiros
# no sistema de ficheiros distribuídos (gluster). Esta diretoria será Mounted em
# cada nó computacional formando um volume gluster
sudo mkdir /var/sharedfiles
sudo chmod 777 /var/sharedfiles
# Fazer STOP à vm base onde se instalou o software
# Criar imagem "Create new machine image" da vm base para posteriormente criar 3 nós
# computacionais com o mesmo Boot Disk. A partir da imagem podem criar-se novas VM,
# por exemplo: tpa2-node1, tpa2-node2 e tpa2-node3

# As instruções seguintes assumem um cenário em que já existem 3 VM com os nomes
# tpa2-node1, tpa2-node2 e tpa2-node3
# Em cada VM lançar o daemon do Gluster e verificar que está em execução
sudo service glusterd start
# Observar o estado do serviço gluster em execução através do comando:
sudo service glusterd status
# Colocar no ficheiro /etc/hosts das 3 Vm a associação de nomes dos nodes e
# os IP internos dos 3 nós computacioanais
10.128.0.8 tpa2-node1
10.128.0.10 tpa2-node2
10.128.0.11 tpa2-node3
# (ATENÇÃO: Só executar num node, ex: tpa2-node1): Definir os nós peers
sudo gluster peer probe tpa2-node2
sudo gluster peer probe tpa2-node3
# Em cada nó verificar que já são gluster peers dos outros nodes
sudo gluster peer status
# (ATENÇÃO: Só executar num node, ex: tpa2-node1): Criar um Gluster volume
sudo gluster volume create glustervol replica 3 tpa2-node1:/var/gluster/brick \
tpa2-node2:/var/gluster/brick tpa2-node3:/var/gluster/brick force
# No mesmo node onde se criou o volume, por exemplo no tpa2-node1
sudo gluster volume start glustervol
# Em cada node associar (mount) a diretoria /var/sharedfiles para o gluster volume
# Por exemplo para o nó tpa2-node1
sudo mount -t glusterfs tpa2-node1:/glustervol /var/sharedfiles
# Note que não é possível fazer mount do gluster brick como diretoria partilhada
sudo mount -t glusterfs tpa2-node1:/glustervol /var/gluster/brick

# Para testar que todos os nós partiham um volume com réplicas de ficheiros
# execute numa das Vms o comando linux
date > /var/sharedfiles/date.txt
# verifique que o ficheiro está replicado nas 3 Vms na diretoria /var/sharedFiles
# Altere o ficheiro num dos nós e verifique que a alteração existe nas réplicas
```

Anexo 4: 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>2.13.2</version>
</dependency>
```