# Relatório Final - Sistemas Distribuídos

## 1. Introdução

Este relatório detalha a implementação de um serviço remoto baseado na arquitetura cliente-servidor, utilizando o protocolo requisição-resposta via sockets UDP e representando as mensagens em formato JSON. O objetivo do projeto é aplicar conceitos de sistemas distribuídos, como serialização, comunicação via rede e tratamento de falhas.

## 2. Arquitetura do Sistema

O sistema foi projetado com base na arquitetura cliente-servidor e segue os princípios de chamada remota a métodos, utilizando Proxy, Esqueleto e Despachante para garantir flexibilidade e modularidade.

Os principais componentes do sistema são:

• \*\*ClienteUDP.java\*\* - Responsável por enviar requisições ao servidor via UDP.

• \*\*ServidorUDP.java\*\* - Recebe requisições, processa e envia respostas.

• \*\*Despachante.java\*\* - Identifica o método correto a ser chamado dinamicamente.

• \*\*Servente.java\*\* - Implementa os métodos remotos de cálculo.

• \*\*Envelope.java & Mensagem.java\*\* - Modelos para serialização e estrutura das mensagens em JSON.

## 3. Métodos Implementados

O serviço remoto implementa três métodos principais para cálculo distribuído:

### 3.1 calcularSoma(List<Integer> numeros)

• \*\*Descrição:\*\* Recebe uma lista de números inteiros e retorna a soma dos valores.

• \*\*Entrada:\*\* Lista de números inteiros.

• \*\*Saída:\*\* Um inteiro representando a soma.

• \*\*Exemplo de Entrada JSON:\*\* {"numeros": [10, 20, 30]}

• \*\*Exemplo de Resposta:\*\* "Resultado da soma: 60"

### 3.2 calcularMedia(List<Integer> numeros)

• \*\*Descrição:\*\* Calcula a média dos valores informados.

• \*\*Entrada:\*\* Lista de números inteiros.

• \*\*Saída:\*\* Um número decimal representando a média.

• \*\*Exemplo de Entrada JSON:\*\* {"numeros": [10, 20, 30, 40]}

• \*\*Exemplo de Resposta:\*\* "Resultado da média: 25.0"

### 3.3 resolverEquacaoSegundoGrau(int a, int b, int c)

• \*\*Descrição:\*\* Resolve uma equação do segundo grau na forma ax² + bx + c = 0.

• \*\*Entrada:\*\* Três coeficientes inteiros (a, b e c).

• \*\*Saída:\*\* As raízes da equação ou uma mensagem indicando que não há raízes reais.

• \*\*Exemplo de Entrada JSON:\*\* {"a": 1, "b": -3, "c": 2}

• \*\*Exemplo de Resposta:\*\* "Raízes da equação: x1=2, x2=1"

## 4. Comunicação Cliente-Servidor

O sistema utiliza comunicação baseada em sockets UDP para troca de mensagens entre Cliente e Servidor. Cada requisição é serializada em JSON e enviada via DatagramPackets.

### 4.1 Funcionamento da Comunicação

1. O ClienteUDP serializa a requisição em JSON e a envia via socket UDP para o servidor.

2. O ServidorUDP recebe a mensagem, desserializa e repassa ao Despachante.

3. O Despachante utiliza reflexão para chamar o método correto no Servente.

4. O Servente executa a lógica e retorna o resultado ao ServidorUDP.

5. O ServidorUDP reenvia a resposta para o ClienteUDP.

6. O ClienteUDP recebe e exibe a resposta na tela.

## 5. Tratamento de Requisições Duplicadas

O servidor mantém um histórico de requisições já processadas para evitar processamento duplicado. Se um requestID já estiver no histórico, a requisição é ignorada.

## 6. Tratamento de Falhas UDP

O ClienteUDP possui um timeout para evitar bloqueios caso a resposta do servidor demore. Se o timeout for atingido, a requisição é retransmitida.

## 7. Testes e Capturas de Tela

As capturas de tela comprovam a execução correta do sistema, mostrando a comunicação Cliente-Servidor e a resposta para cada requisição enviada.

## 8. Conclusão

O projeto foi concluído com sucesso, garantindo a comunicação cliente-servidor via UDP, serialização JSON, e tratamento de falhas de comunicação e duplicação de requisições. A abordagem de Proxy, Esqueleto e Despachante permite escalabilidade e flexibilidade na adição de novos métodos remotos no futuro.