RELATÓRIO — Trabalho 2: Remote Method Invocation (RMI)

Sistemas Distribuídos

Alunos: Matheus Narcizio (494693) - Maria Davila (586054)

1. Introdução

Este relatório descreve o desenvolvimento do Trabalho 2 – RMI, cujo objetivo é reimplementar a Questão 1 do Trabalho 1 utilizando invocação remota de métodos (RMI) e protocolo requisição–resposta.

A comunicação entre cliente e servidor foi estruturada de acordo com o modelo clássico de RMI:

- O cliente envia requisições codificadas.
- O servidor recebe, decodifica e identifica qual método remoto deve ser executado.
- O servidor envia uma resposta após execução local.
- O cliente valida e exibe o resultado.

A representação externa foi implementada através de JSON.

2. Arquitetura Geral

A aplicação segue a arquitetura RMI com os três métodos principais:

- doOperation() no cliente
- getRequest() no servidor
- sendReply() no servidor

As mensagens são estruturadas com:

messageType, requestId, objectReference, methodId, arguments, result e exception.

3. Entidades do Sistema

O modelo contém as seguintes entidades:

Pessoa, Aluno, Instrutor, Funcionario, Visitante e Academia.

Heranças (é-um):

Aluno → Pessoa, Instrutor → Pessoa, Funcionario → Pessoa, Visitante → Pessoa.

Agregações (tem-um):

Academia contém listas de alunos, instrutores, funcionários e visitantes.

Alunos possuem plano de treino, que contém listas de exercícios.

4. Métodos Remotos Implementados:

- cadastrar_aluno
- cadastrar_instrutor
- registrar_visitante
- avaliar_desempenho

Todos acessíveis via RMI utilizando o METHOD_DISPATCHER do servidor.

5. Representação Externa de Dados

Implementada com JSON. Os objetos são convertidos para JSON, codificados em Base64 e enviados como bytes. A desserialização reverte o processo no servidor e no cliente.

6. Passagem por Referência e ValorPassagem por referência:

Implementada com RemoteObjectRef("AcademiaService").

Passagem por valor:

Argumentos enviados como JSON codificado.

7. Implementação do Protocolo Requisição–Resposta

O protocolo é implementado dentro da classe `Message` em **protocolo.py**

8. Funcionamento do Servidor RMI

O servidor abre socket TCP para aceitar uma conexão, recebe requests com get_request, depois usa o dispatcher para localizar o método remoto, executa localmente usando a instancia academia_servidor, empacota com ReplyMessage e envia com send_reply.

9. Funcionamento do Cliente RMI

O cliente conecta ao servidor, depois cria um RemoteObjectRef("AcademiaService"), chama do_operation que vai construir, empacotar e enviar um RequestMessage, que vai aguardar a reposta e desempacotar para exibir o resultado.