UNIVERSIDADE FEDERAL DO CEARÁ
REDES DE COMPUTADORES
SISTEMAS DISTRIBUÍDOS
FRANCISCO JARBAS DOS SANTOS SOUSA

RELATÓRIO: LABORATÓRIO JAVA RPC SEM RMI

QUIXADÁ-CE
2025

Sumário

INTRODUÇÃO

Este laboratório demonstra a implementação de um sistema de Remote Procedure Call (RPC) utilizando Java e sockets, sem o uso do Java RMI (Remote Method Invocation). A abordagem adotada simula chamadas remotas de métodos através de comunicação por sockets, permitindo a interação entre um cliente e um servidor.

CONFIGURAÇÃO DO PROJETO

O projeto é estruturado como um projeto Maven, com um arquivo pom.xml básico contendo as configurações essenciais, como o Group ID, Artifact ID e versão.

Exemplo de pom.xml:

```
http://maven.apache.org/xsd/maven4.0.0.xsd (xsi:schemaLocation)
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.exemplo</groupId>
    <artifactId>JavaRPC-Sockets</artifactId>
    <version>1.0-SNAPSHOT</version>
</project>
```

DESENVOLVIMENTO DO CÓDIGO

A implementação segue a seguinte estrutura:

- Interface RPC (HelloService.java)
- Implementação do Serviço (HelloServiceImpl.java)
- Servidor (Server.java)
- Cliente (Client.java)

INTERFACE RPC

A interface HelloService define o contrato do serviço RPC. O método sayHello(String name) é declarado para ser chamado remotamente.

```java
package com.exemplo;

// Interface do serviço
public interface HelloService {
    String sayHello(String name);
}
```

IMPLEMENTAÇÃO DO SERVIÇO

A classe HelloServiceImpl implementa a interface HelloService, fornecendo uma lógica simples para responder às requisições.

```java
package com.exemplo;

// Implementação do serviço
public class HelloServiceImpl implements HelloService {
    @Override
    public String sayHello(String name) {
        return "Olá, " + name + "! Este é um exemplo de RPC com sockets.";
    }
}
```

IMPLEMENTAÇÃO DO SERVIDOR

O servidor escuta conexões na porta 5000, processa requisições e retorna respostas conforme a chamada remota.

```java
package com.exemplo;

import java.io.*;

public class Server {
    public static void main(String[] args) {
        try (ServerSocket serverSocket = new ServerSocket(5000)) {
            System.out.println("Servidor pronto na porta 5000...");

            // Espera por conexões
            while (true) {
                Socket clientSocket = serverSocket.accept();
                System.out.println("Cliente conectado!");
                try (BufferedReader in = new BufferedReader(new InputStreamReader(clientSocket.getInputStream()));
                        PrintWriter out = new PrintWriter(clientSocket.getOutputStream(), true)) {

                    // Recebe a requisição
                    String input = in.readLine();
                    System.out.println("Requisição recebida: " + input);
                    // Processa a chamada ao método
                    if (input.startsWith("sayHello:")) {
                        String name = input.split(":")[1];
                        HelloService service = new HelloServiceImpl();
                        String response = service.sayHello(name);
                        // Envia a resposta de volta ao cliente
                        out.println(response);
                    } else {
                        out.println("Método não suportado.");
                    }
                }
                clientSocket.close();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
```

IMPLEMENTAÇÃO DO CLIENTE

O cliente estabelece uma conexão com o servidor e faz a chamada RPC enviando uma string formatada.

```java
package com.exemplo;

import java.io.*;

public class Client {
    public static void main(String[] args) {
        try (Socket socket = new Socket("localhost", 5000);
                BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()));
                PrintWriter out = new PrintWriter(socket.getOutputStream(), true)) {

            // Envia a chamada do método para o servidor
            String request = "sayHello:Usuário";
            out.println(request);
            System.out.println("Requisição enviada: " + request);
            // Recebe a resposta do servidor
            String response = in.readLine();
            System.out.println("Resposta do servidor: " + response);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

CONCLUSÃO

O laboratório demonstra como é possível implementar um sistema de chamadas remotas usando sockets em Java, sem depender de RMI. O modelo adotado é simples, mas eficaz para entender conceitos básicos de RPC. No entanto, em sistemas reais, protocolos mais robustos como gRPC ou RESTful APIs são geralmente preferidos para maior escalabilidade e segurança.