**Título:** adaptar conforme o tema do trabalho:  **SISTEMAS DISTRIBUÍDOS: observações sobre a implementação de RMI na plataforma Windows utilizando a linguagem Java**

**Alunos:** nome completo dos membros do grupo.

1. **Introdução** - Escrever sobre  o problema, objetivos e metodologia (qual configuração de hardware e software usará, quais programas(processos) rodarão, o que farão, por quanto tempo, em detalhes ),  até 1 página.

**2- Revisão Bibliográfica –**

Escrever sobre as ferramentas ou recursos usados na implementação, focando na linguagem escolhida, API ou biblioteca, midleware, etc. adotados, fazendo as devidas citações, de livros e páginas web, conforme norma ABNT (autor, data) entre 1 e 2 páginas.

Para este trabalho, além da linguagem java, foram utilizadas algumas bibliotecas para auxílio na programação com RMI, a saber:

java.net.MalformedURLException

De acordo com essa classe lança uma exceção quando uma URL é informada de maneira incorreta;

As classes listadas a seguir tem suas definições de acordo com Jav,,,, e não são obrigatórias em métodos principais de manipulação de RMI.

java.rmi.registry.LocateRegistry;

LocateRegistry é usado para obter uma referência a um registro de objeto remoto de bootstrap em um host específico (incluindo o host local) ou para criar um registro de objeto remoto que aceita chamadas em uma porta específica.

java.rmi.registry.Registry;

Registryé uma interface remota para um registro de objeto remoto simples que fornece métodos para armazenar e recuperar referências de objeto remoto vinculadas a nomes de string arbitrários. Os métodos bind, unbinde rebind são usados ​​para alterar as ligações de nomes no registro, e os métodos lookupe listsão usados ​​para consultar as ligações de nomes atuais.

Em seu uso típico, a Registrypermite a inicialização do cliente RMI: fornece um meio simples para um cliente obter uma referência inicial a um objeto remoto. Portanto, a implementação de objeto remoto de um registro normalmente é exportada com um endereço conhecido, como um ObjIDnúmero de porta TCP conhecido e conhecido (o padrão é 1099).

A LocateRegistryclasse fornece uma API programática para construir uma referência de bootstrap para um Registryendereço remoto (consulte os getRegistrymétodos estáticos) e para criar e exportar um Registryna VM atual em um endereço local específico (consulte os createRegistrymétodos estáticos).

Uma Registryimplementação pode optar por restringir o acesso a alguns ou a todos os seus métodos (por exemplo, métodos que alteram as ligações do registro podem ser restritos a chamadas originadas do host local). Se um Registry método optar por negar acesso para uma determinada invocação, sua implementação pode lançar AccessException, que (porque estende RemoteException) será encapsulado em a ServerExceptionquando capturado por um cliente remoto.

java.rmi.Remote;

A Remoteinterface serve para identificar interfaces cujos métodos podem ser invocados de uma máquina virtual não local. Qualquer objeto remoto deve implementar direta ou indiretamente essa interface. Apenas os métodos especificados em uma "interface remota", uma interface que se estende java.rmi.Remote, estão disponíveis remotamente.

As classes de implementação podem implementar qualquer número de interfaces remotas e podem estender outras classes de implementação remota. O RMI fornece algumas classes de conveniência que as implementações de objetos remotos podem estender, facilitando a criação de objetos remotos. Essas classes são java.rmi.server.UnicastRemoteObjecte java.rmi.activation.Activatable.

Os nomes usados ​​para ligações em a Registrysão strings puras, não analisadas. Um serviço que armazena sua referência remota em um Registrypode desejar usar um nome de pacote como um prefixo na vinculação de nomes para reduzir a probabilidade de colisões de nomes no registro.

import java.rmi.server.UnicastRemoteObject;

Usado para exportar um objeto remoto com JRMP e obter um stub que se comunica com o objeto remoto. Os stubs são gerados em tempo de execução usando objetos proxy dinâmicos ou são gerados estaticamente em tempo de compilação, normalmente usando a rmicferramenta.

As classes listadas a seguir tem suas definições de acordo com Jav,,,, e **são** obrigatórias em métodos principais de manipulação de RMI.

java.rmi.NotBoundException

Lança uma exceção se for feita uma tentativa de pesquisar ou desvincular no registro um nome que não tenha sido associado anteriormente;

java.rmi.RemoteException;

a superclasse comum para várias exceções relacionadas à comunicação que podem ocorrer durante a execução de uma chamada de método remoto. Cada método de uma interface remota, uma interface que estende java.rmi.Remote, deve listar RemoteExceptionem sua cláusula throws.

java.rmi.server.ServerNotActiveException;

Lança uma exceção durante uma chamada para RemoteServer.getClientHost se o método getClientHost for chamado fora de uma chamada de método remoto em serviço.

**3- Implementação -** Escrever sobre a implementação, para que serve, explicando o código-fonte (colar o código-fonte e print das telas, explicá-las e colocar legenda) e software ou hardware de apoio, entre 1 e 2 páginas.

**4- Análise dos Resultados -**Escrever sobre as observações e análises, em relação à execução do sistema (colar print das telas, explicá-las e  colocar legenda). Entre 1 e 2 páginas.

**5 – Referências -**Listar todos os  autores (livros, artigos, sites) citados ou adotados, conforme norma ABNT.

ORACLE. **Java™ Platform, Standard Edition 7 -** API Specification. Disponível em <https://docs.oracle.com/javase/7/docs/api/java/rmi> . Acesso em 09/Abr/2022.

ORACLE. **Class MalformedURLException**. Disponível em <https://docs.oracle.com/javase/8/docs/api/java/net/MalformedURLException.html> . Acesso em 09/Abr/2022.