Missão Prática | Nível 5 | Mundo 3

Por que não paralelizar

Rian Joseph Ramos Felizardo - 202202923931

POLO BARREIRO - Belo Horizonte, MG

Nível 5 - **Vamos integrar sistemas** – 2023.1 – 3° Semestre Letivo

Repositório - [desenvolvimento-sistemas-mundo-3/nivel-04/src/CadastroEE at main · rianjsp/desenvolvimento-sistemas-mundo-3]

Objetivo da Prática

Descreva nessa seção qual o objetivo da sua prática. Todos os Relatórios de Práticas deverão ser confeccionados em arquivo no formato PDF, com a **Logo da Universidade**, **nome do Campus**, **nome do Curso**, **nome da Disciplina**, **número da Turma**, **semestre letivo**. Além disso, o projeto deve ser armazenado em um repositório no GIT e o respectivo endereço deve constar na documentação e essa documentação deve estar no GIT. O código deve estar versionado no GIT de forma organizada.

Lembre-se que a organização contará pontos.

OBJETIVOS

1. Criar servidores Java com base em Sockets.
2. Criar clientes síncronos para servidores com base em Sockets.
3. Criar clientes assíncronos para servidores com base em Sockets.
4. Utilizar Threads para implementação de processos paralelos.
5. No final do exercício, o aluno terá criado um servidor Java baseado em Socket, com acesso ao banco de dados via JPA, além de utilizar os recursos nativos do Java para implementação de clientes síncronos e assíncronos. As Threads serão usadas tanto no servidor, para viabilizar múltiplos clientes paralelos, quanto no cliente, para implementar a resposta assíncrona.

1º Procedimento | Criando o Servidor e Cliente de Teste

Inserir neste campo, **de forma organizada**, todos os códigos do roteiro do 1º Procedimento da Atividade Prática, os resultados da execução do código e a Análise e Conclusão:

Análise e Conclusão:

- **Como funcionam as classes Socket e ServerSocket?**

As classes Socket e ServerSocket em Java são usadas para comunicação cliente-servidor via TCP/IP.

Em geral, o ServerSocket é usado para esperar e aceitar conexões de clientes, enquanto o Socket é usado pelo cliente para se conectar ao servidor e trocar dados.

- **Qual a importância das portas para a conexão com servidores?**

As portas são essenciais para a comunicação em redes porque permitem que múltiplas aplicações em um mesmo servidor se comuniquem de forma independente, utilizando a mesma máquina. Elas funcionam como canais lógicos, direcionando o tráfego para os processos corretos.

- **Para que servem as classes de entrada e saída ObjectInputStream e ObjectOutputStream, e por que os objetos transmitidos devem ser serializáveis?**

Em geral, as classes de entrada e saída ObjectInputStream e ObjectOutputStream permitem a troca de objetos entre sistemas, e os objetos precisam ser serializáveis para que possam ser convertidos em bytes e transmitidos de forma segura e eficiente.

- **Por que, mesmo utilizando as classes de entidades JPA no cliente, foi possível garantir o isolamento do acesso ao banco de dados?**

O isolamento do acesso ao banco de dados foi garantido no cliente, mesmo utilizando as classes de entidades JPA, porque o cliente não acessa diretamente o banco de dados. Ele interage com o servidor que encapsula toda a lógica de acesso ao banco por meio de controladores JPA.

A JPA no cliente é usada apenas para comunicar e enviar dados, mas o **acesso real ao banco de dados** ocorre no servidor, garantindo o **isolamento** e a **segurança** das operações no banco.

Observe que os tópicos acima seguem exatamente o que está na Atividade Prática exigida.

2º Procedimento | Servidor Completo e Cliente Assíncrono

Inserir neste campo, **de forma organizada**, todos os códigos do roteiro do 2º Procedimento da Atividade Prática, os resultados da execução do código e a Análise e Conclusão:

Análise e Conclusão:

- **Como as Threads podem ser utilizadas para o tratamento assíncrono das respostas enviadas pelo servidor?**

As **Threads** podem ser usadas para o tratamento assíncrono das respostas enviadas pelo servidor.

Usar **Threads** para tratar respostas do servidor de forma assíncrona permite que o cliente receba e processe dados enquanto continua executando outras tarefas, **sem bloquear o fluxo principal** da aplicação.

- **Para que serve o método invokeLater, da classe SwingUtilities?**

O **invokeLater** garante que a atualização da interface gráfica seja realizada **na thread correta**, prevenindo erros de concorrência e garantindo a integridade da UI.

- **Como os objetos são enviados e recebidos pelo Socket Java?**

Envio: O objeto é serializado e enviado usando ObjectOutputStream.

Recebimento: O objeto é desserializado e recebido usando ObjectInputStream.

Ambos os objetos precisam implementar Serializable para serem transmitidos via Socket.

- **Compare a utilização de comportamento assíncrono ou síncrono nos clientes com Socket Java, ressaltando as características relacionadas ao bloqueio do processamento:**

Síncrono: Mais simples, mas pode levar a bloqueios e ineficiência, pois o cliente espera pela resposta do servidor antes de continuar.

Assíncrono: Mais complexo, mas permite maior flexibilidade e uso eficiente dos recursos, pois o cliente pode continuar executando outras tarefas enquanto aguarda a resposta.

Observe que os tópicos acima seguem exatamente o que está na Atividade Prática exigida.