Curso: Desenvolvimento Full Stack
Disciplina: Por que paralelizar?
Turma: 9001
Semestre: 3°
Aluno: Elmar G. Lorena
Endereço Repositório Github:
https://github.com/PerfilEstudos/Nivel5_Mundo3.git

1. **Título da Prática** – Criando um servidor Java baseado em Socket.

Objetivo da Prática – Desenvolver 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.

2. **Todos os códigos solicitados neste roteiro de aula** – Segue link do arquivo postado em https://github.com/PerfilEstudos/Nivel5_Mundo3.git

3. **Os resultados da execução dos códigos também devem ser apresentados;**

4. **Análise e Conclusão:**

   A. **Como funcionam as classes Socket e ServerSocket?** Ao contrário da classe ServerSocket que funciona como um Servidor escutando o cliente, a classeSocket é o cliente propriamente dito. Socket provê a comunicação entre duas pontas (fonte e destino) - também conhecido como two-way communication - entre dois processos que estejam na mesma máquina (Unix Socket) ou na rede (TCP/IP Sockets).

   B. **Qual a importância das portas para a conexão com servidores?** As portas permitem que os computadores diferenciem facilmente entre diferentes

tipos de tráfego: os e-mails vão para uma porta diferente daquela das páginas web.

C. **Para que servem as classes de entrada e saída ObjectInputStream e ObjectOutputStream, e por que os objetos transmitidos devem ser serializáveis?** As classes chamadas ObjectInputStream e ObjectOutputStream são responsáveis por inserir e recuperar objetos serializados do stream.

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

E. **Como as Threads podem ser utilizadas para o tratamento assíncrono das respostas enviadas pelo servidor?** Operações assíncronas garantem que a interface não fique travada durante esses processos. Microservices e APIs: A programação assíncrona facilita a escalabilidade em arquiteturas de microsserviços, permitindo que serviços lidem com solicitações de forma independente, melhorando a eficiência geral do sistema.

F. **Para que serve o método invokeLater, da classe SwingUtilities?** A classe SwingUtilities, presente no conjunto de bibliotecas do Java Swing, oferece métodos úteis para lidar com eventos em interfaces gráficas. No contexto da questão, destaca-se o método invokeLater()

G. **Como os objetos são enviados e recebidos pelo Socket Java?** São utilizadas em uma aplicação com Socket (Java). Os dados que serão enviados e recebidos pelo servidor passarão todos através deste socket do antes de chegar ao usuário de forma tratada.

H. **Compare a utilização de comportamento assíncrono ou síncrono nos clientes com Socket Java, ressaltando as características relacionadas ao bloqueio do processamento.** Ao executar algo de forma síncrona, você espera que termine antes de passar para outra tarefa. Ao executar algo de forma assíncrona, você pode passar para outra tarefa antes de terminar.