Nível 5: Por que não paralelizar?

Ana Clara Quidute Marinho Silva - 202302798691

Campus Polo Setor Central - Araguaína(TO)

Desenvolvimento Full-Stack - Turma 2023.1 - Semestre Letivo 2024.1

Objetivo da prática:

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

Análises e Conclusões

a. As classes Socket e ServerSocket são usadas para a comunicação de rede usando o protocolo TCP. Enquanto a classe Socket é responsável por criar a conexão, a classe ServerSocket é responsável por criar o servidor que aguarda a conexão.
b. As portas são importantes para a conexão com servidores porque elas funcionam como pontos de acesso específicos nos computadores para a troca de dados, sendo essenciais para organizar e gerenciar a comunicação entre diferentes serviços em uma rede.
c. Protocolos de comunicação, como o TCP, operam em níveis de dados binários. Serializar objetos transforma estrutura de dados complexas em bytes, por isso os objetos transmitidos devem ser serializados, e a serialização e desserialização ocorrem através das classes ObjectInputStream e ObjectOutputStream.
d. Devido à arquitetura e ao padrão de projeto adotados na aplicação.

2º Procedimento | Servidor Completo e Cliente Assíncrono

Análise e conclusões:

a. Uma forma de utilizar Threads para o tratamento assíncrono das respostas enviadas pelo servidor é criar uma classe que implemente a interface Runnable, onde você define o que a Thread deve executar. Em seguida, você cria uma instância da classe Thread, passando a instância de Runnable, e chama o método start() para iniciar a thread. Isso permite que a execução do código definido no método run() da classe Runnable ocorra em paralelo ao fluxo principal da aplicação.
b. O método invokeLater da classe SwingUtilities serve para garantir que um determinado bloco de código seja executado na thread responsável por manipular a interface gráfica do usuário (GUI) no Swing, assim garantindo a consistência e segurança em aplicações gráficas.
c. Os objetos são enviados e recebidos através de sockets utilizando as classes ObjectOutputStream e ObjectInputStream, as classes que permitem a serialização e desserialização, como vimos na análise no 1º procedimento.
d. No comportamento assíncrono as operações de E/S são realizadas sem bloqueio, permitindo que o cliente execute outras tarefas enquanto aguarda a resposta do servidor, logo a implementação é mais complexa, mas usa os recursos de forma eficiente. Assim, caracterizando o comportamento assíncrono como ideal para aplicações que demanda alta performance, responsividade e precisa escalar para muitos clientes simultâneos;
   No comportamento síncrono as operações de E/S bloqueiam o fluxo de execução até que a operação seja concluída, mas, diferente do comportamento assíncrono, a implementação é direta, menos complexa.Por isso o comportamento síncrono é ideal para aplicações simples ou onde não se justifica uma alta complexidade no gerenciamento de threads.