Relatório da Missão Prática - Nível 5

Universidade Estácio de Sá
Curso: Tecnologia em Desenvolvimento Full Stack
Disciplina: RPG0018 - Por que não paralelizar?
Aluno: Carlos Alexandre Paulino de Oliveira
Matrícula: 202308511361

Objetivos da Prática

O objetivo desta prática é desenvolver habilidades na criação de sistemas de comunicação distribuída em Java, utilizando sockets para estabelecer a conexão entre servidores e clientes, e explorar o uso de Threads para permitir operações paralelas tanto no lado do servidor quanto no lado do cliente.

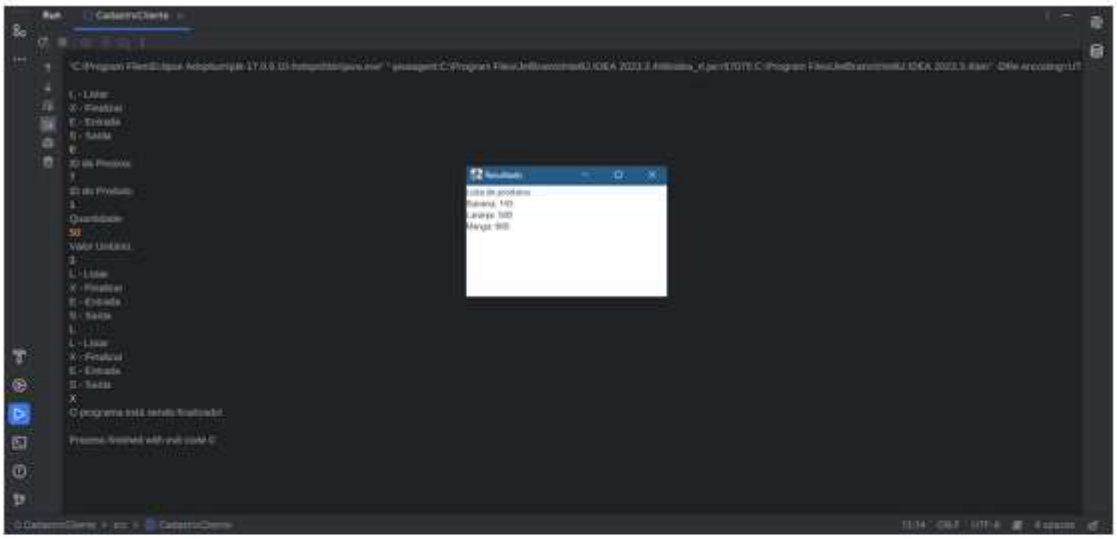
Os objetivos específicos incluem criar um servidor Java capaz de receber conexões de clientes, validar credenciais de login e senha, responder a comandos do cliente e acessar um banco de dados SQL Server via JPA. Além disso, visa-se implementar um cliente de teste para interagir com o servidor de forma síncrona, enviando comandos e recebendo respostas, e ampliar o servidor para lidar com operações de entrada e saída de produtos, permitindo a interação assíncrona com o cliente.

Também será desenvolvido um cliente assíncrono que permita interações dinâmicas por meio de uma interface de linha de comando e uma janela de mensagens, demonstrando o uso de Threads para tratamento assíncrono de respostas do servidor. Adicionalmente, serão exploradas as funcionalidades do NetBeans para aumentar a produtividade no desenvolvimento do sistema e será realizada uma reflexão sobre o uso de Threads e sockets em ambientes distribuídos, analisando os resultados obtidos e tirando conclusões sobre a eficácia do modelo adotado.

Resultados

1º Procedimento: Criando o servidor e cliente de teste .

2º Procedimento: Servidor completo e cliente assíncrono .



Análise e Conclusão

1. Como funcionam as classes Socket e ServerSocket?

As classes Socket e ServerSocket em Java são usadas para criar conexões de rede entre um cliente e um servidor. O ServerSocket é usado pelo servidor para aguardar e aceitar solicitações de conexão dos clientes. Uma vez que uma conexão é estabelecida, um objeto Socket é criado no servidor para representar essa conexão. O Socket é usado pelo servidor para se comunicar com o cliente, enviando e recebendo dados.

2. Qual a importância das portas para a conexão com servidores?

As portas são fundamentais para a conexão com servidores porque permitem que vários serviços de rede sejam executados em um mesmo computador, cada um em sua própria

porta. Ao conectar-se a um servidor, o cliente especifica a porta na qual o serviço desejado está sendo executado, permitindo que o servidor direcione os dados para o serviço correto.

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

As classes ObjectInputStream e ObjectOutputStream são usadas para ler e escrever objetos Java a partir de e para fluxos de dados. Os objetos transmitidos devem ser serializáveis para serem convertidos em uma sequência de bytes que podem ser enviados pela rede.

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

O acesso ao banco de dados é realizado exclusivamente no servidor. O cliente interage com o servidor através de solicitações e respostas, mas não acessa diretamente o banco de dados. Isso garante o isolamento, a segurança e a integridade dos dados.

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

As Threads podem ser utilizadas criando-se Threads separadas para lidar com cada solicitação de cliente. Isso permite que o servidor processe solicitações de forma paralela e responda de maneira assíncrona, garantindo eficiência e escalabilidade.

6. Para que serve o método invokeLater, da classe SwingUtilities?

O método invokeLater da classe SwingUtilities é usado para executar uma tarefa de forma assíncrona na thread de despacho de eventos do Swing. Ele é essencial para atualizar a interface gráfica do usuário de forma segura.

7. Como os objetos são enviados e recebidos pelo Socket Java?

Os objetos são serializados em uma sequência de bytes antes de serem enviados pela rede. No lado receptor, esses bytes são deserializados de volta ao objeto original.

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

O comportamento assíncrono permite operações simultâneas, garantindo maior responsividade, enquanto o comportamento síncrono bloqueia o processamento até a conclusão de uma operação, sendo mais adequado para tarefas rápidas e sequenciais.