



***Campus: Estácio de Sá Porto Velho Polo São Cristóvão***

***Curso: Desenvolvimento Full Stack***

***Disciplina: Nível 5: Por Que Não Paralelizar?***

***Semestre: 2024.4 Full Stack***

***Integrante: Ismael Lucena de Albuquerque***

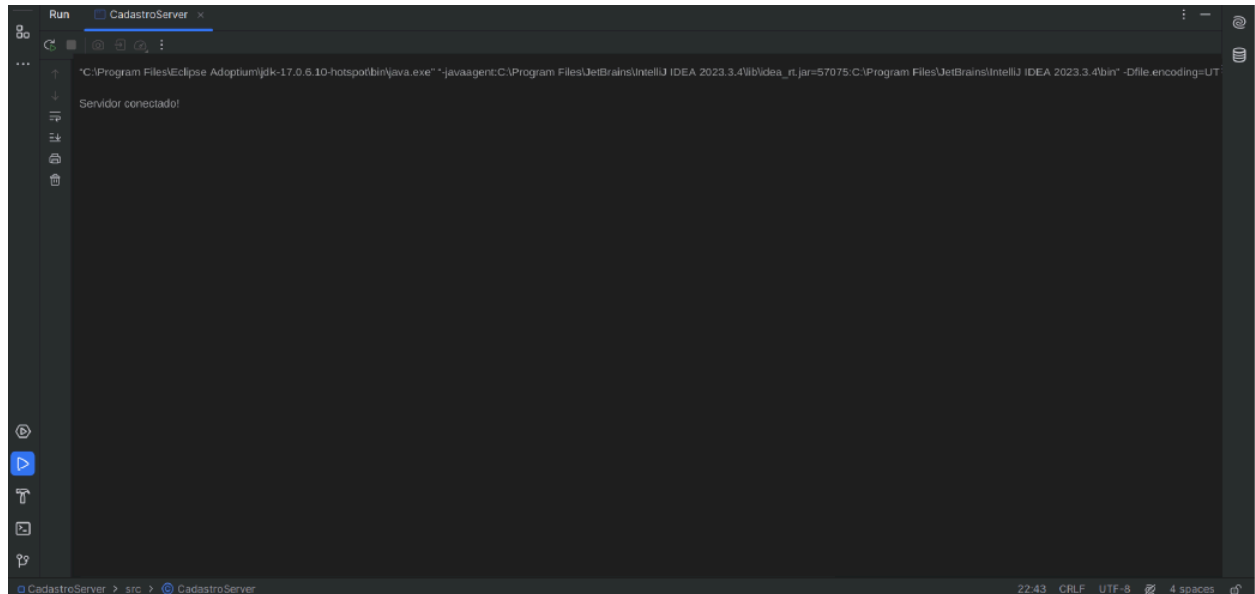
***Número de Matrícula: 202311173691***

### ***Objetivos 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.*

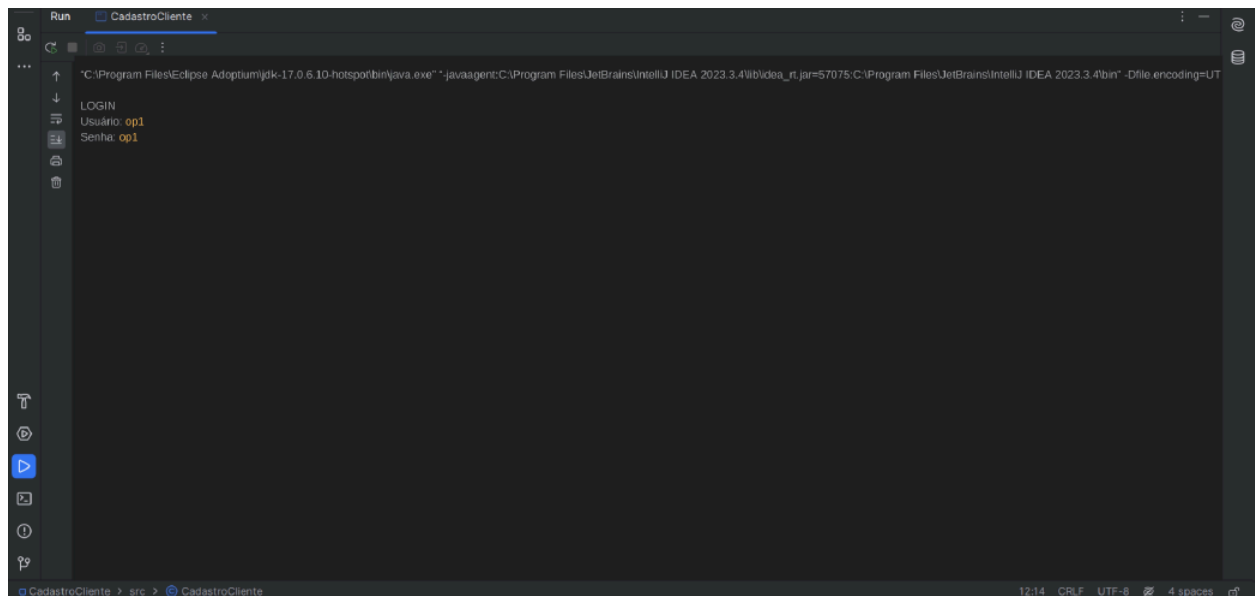
# Resultados:

## 1º Procedimento | Criando o Servidor e Cliente de Teste



The screenshot shows the 'Run' console for the 'CadastroServer' application. The console output displays the full Java command used to run the application, followed by the message 'Servidor conectado!'. The status bar at the bottom indicates the file encoding is UTF-8 and the line separator is CRLF.

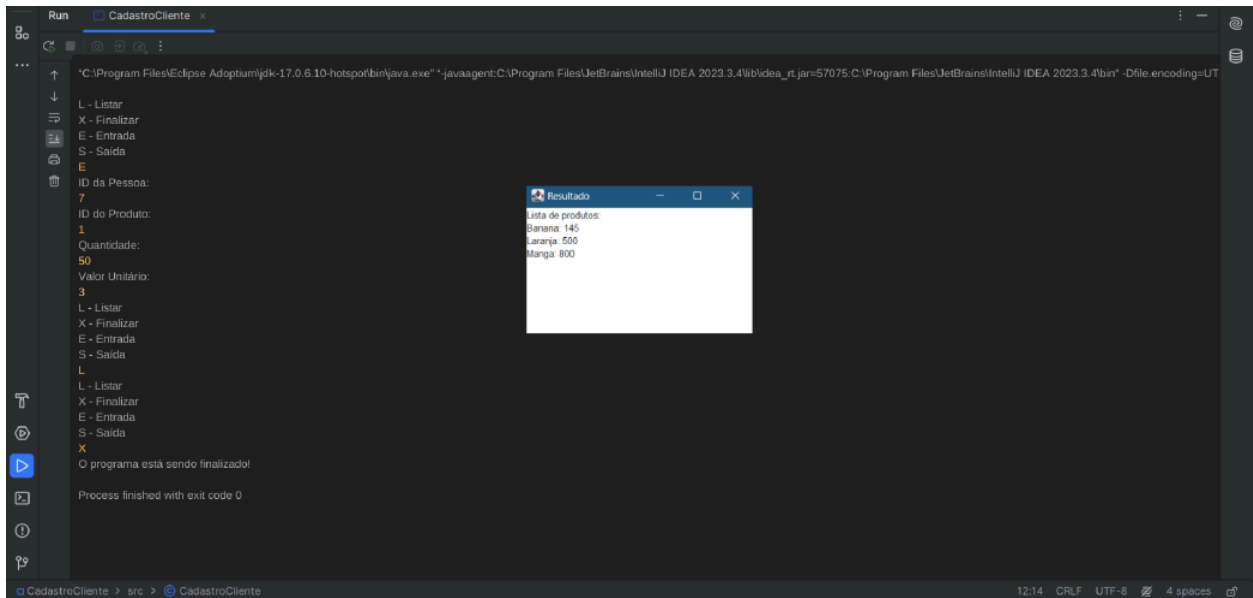
```
Run CadastroServer x
...
"C:\Program Files\Eclipse Adoptium\jdk-17.0.6.10-hotspot\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.4\lib\idea_rt.jar=57075:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.4\bin" -Dfile.encoding=UTF-8
Servidor conectado!
CadastroServer > src > CadastroServer
22:43 CRLF UTF-8 4 spaces
```



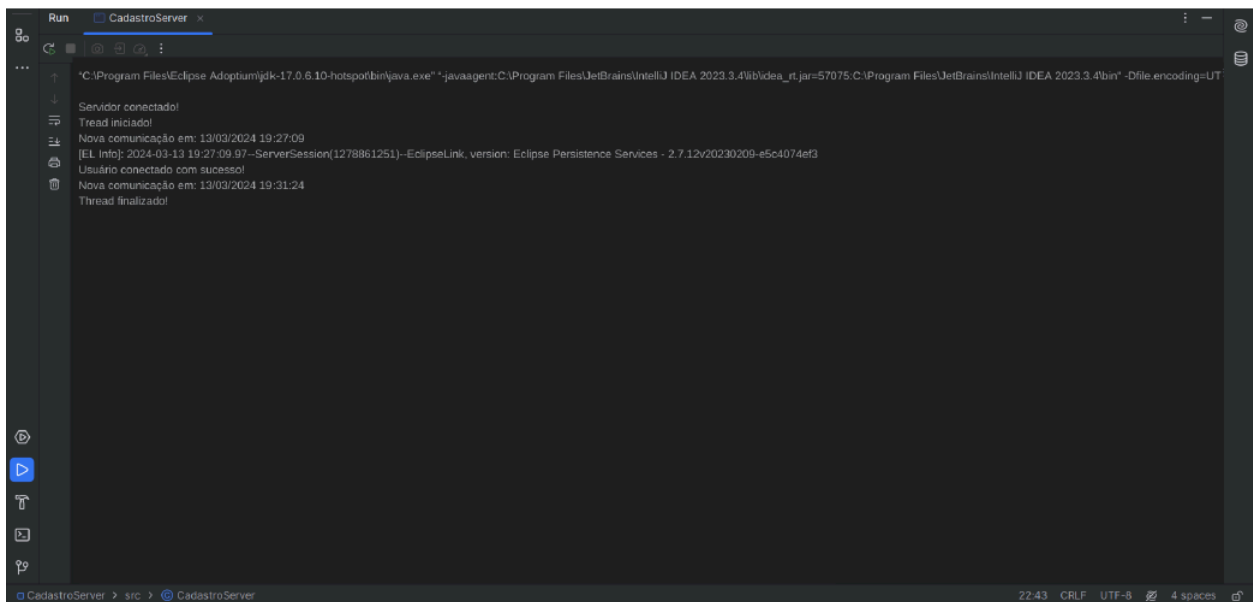
The screenshot shows the 'Run' console for the 'CadastroCliente' application. The console output displays the full Java command used to run the application, followed by the message 'LOGIN' and the input values 'Usuário: op1' and 'Senha: op1'. The status bar at the bottom indicates the file encoding is UTF-8 and the line separator is CRLF.

```
Run CadastroCliente x
...
"C:\Program Files\Eclipse Adoptium\jdk-17.0.6.10-hotspot\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.4\lib\idea_rt.jar=57075:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.4\bin" -Dfile.encoding=UTF-8
LOGIN
Usuário: op1
Senha: op1
CadastroCliente > src > CadastroCliente
12:14 CRLF UTF-8 4 spaces
```

## 2º Procedimento | Servidor Completo e Cliente Assíncrono



```
Run CadastroCliente x
"C:\Program Files\Eclipse Adoptium\jdk-17.0.6-hotspot\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.4\idea_rt.jar=57075:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.4\bin" -Dfile.encoding=UTF-8
L - Listar
X - Finalizar
E - Entrada
S - Saida
E
ID da Pessoa:
7
ID do Produto:
1
Quantidade:
50
Valor Unitário:
3
L - Listar
X - Finalizar
E - Entrada
S - Saida
L
L - Listar
X - Finalizar
E - Entrada
S - Saida
X
O programa está sendo finalizado!
Process finished with exit code 0
```



```
Run CadastroServer x
"C:\Program Files\Eclipse Adoptium\jdk-17.0.6-hotspot\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.4\idea_rt.jar=57075:C:\Program Files\JetBrains\IntelliJ IDEA 2023.3.4\bin" -Dfile.encoding=UTF-8
Servidor conectado!
Tread iniciado!
Nova comunicação em: 13/03/2024 19:27:09
[EL Info]: 2024-03-13 19:27:09.97--ServerSession(1278861251)--EclipseLink, version: Eclipse Persistence Services - 2.7.12v20230209-e5c4074ef3
Usuário conectado com sucesso!
Nova comunicação em: 13/03/2024 19:31:24
Thread finalizado!
```

# Análise e Conclusão

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

As Threads permitem que o servidor processe múltiplas conexões de clientes simultaneamente, atribuindo uma Thread específica para gerenciar cada cliente. Isso garante que o servidor possa responder de forma assíncrona a diferentes clientes sem bloquear as requisições uns dos outros. No cliente, Threads podem ser usadas para ouvir respostas do servidor enquanto a interface gráfica ou outras operações continuam em execução.

## **2 Para que serve o método `invokeLater`, da classe `SwingUtilities`?**

O método `SwingUtilities.invokeLater` é usado para executar atualizações de componentes gráficos na Thread de eventos do Swing. Essa abordagem garante a integridade dos dados e evita condições de corrida, já que o Swing não é thread-safe. É especialmente útil em aplicações que precisam realizar atualizações na interface gráfica com base em dados recebidos de outras Threads.

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

Os objetos são enviados e recebidos através das classes `ObjectOutputStream` e `ObjectInputStream`. Antes do envio, os objetos precisam ser serializáveis, permitindo que sejam convertidos em uma sequência de bytes para transporte pela rede. No lado receptor, o fluxo de entrada reconstrói os objetos a partir desses bytes, garantindo que a comunicação entre cliente e servidor seja eficiente e segura.

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

No comportamento síncrono, o cliente envia uma requisição e bloqueia até que receba uma resposta do servidor. Esse modelo é mais simples de implementar, porém pode resultar em espera ativa, prejudicando a experiência do usuário em sistemas de alta latência ou com múltiplas operações.

No comportamento assíncrono, o cliente pode continuar executando outras tarefas enquanto aguarda a resposta do servidor. Esse modelo utiliza Threads ou mecanismos de callback para tratar as respostas quando chegam, evitando o bloqueio do processamento. É mais adequado para sistemas interativos e de alta performance, mas exige uma implementação mais complexa para gerenciar a concorrência.

