



# Estácio

**Atividade:** Relatório de Prática

**Nome:** Jamison Queiroz

**Matrícula:** 202208101127

**Curso:** Desenvolvimento Full Stack

**Turma:** 2022.4

**Disciplina:** Por Que Não Paralelizar?

**Professor:** Guilherme Dutra Gonzaga Jaime

## Sumário

|  |          |
|--|----------|
| <b>1. INTRODUÇÃO .....</b>                       | <b>3</b> |
| <b>2. OBJETIVO .....</b>                         | <b>3</b> |
| <b>3. SOFTWARE UTILIZADO.....</b>                | <b>3</b> |
| <b>4. PROCEDIMENTOS.....</b>                     | <b>3</b> |
| 4.1 Criando o Servidor e Cliente de Teste .....  | 3        |
| 4.1.1 Análise e Conclusão .....                  | 4        |
| 4.2 Servidor Completo e Cliente Assíncrono ..... | 5        |
| 4.2.1 Análise e Conclusão .....                  | 6        |
| <b>5. CONCLUSÃO .....</b>                        | <b>7</b> |

## **1. INTRODUÇÃO**

Nesta prática iremos criar servidores e clientes baseados em Socket, com uso de Threads tanto no lado cliente quanto no lado servidor, acessando o banco de dados via JPA.

## **2. OBJETIVO**

- 2.1.** Criar servidores Java com base em Sockets.
- 2.2.** Criar clientes síncronos para servidores com base em Sockets.
- 2.3.** Criar clientes assíncronos para servidores com base em Sockets.
- 2.4.** Utilizar Threads para implementação de processos paralelos.
- 2.5.** 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.

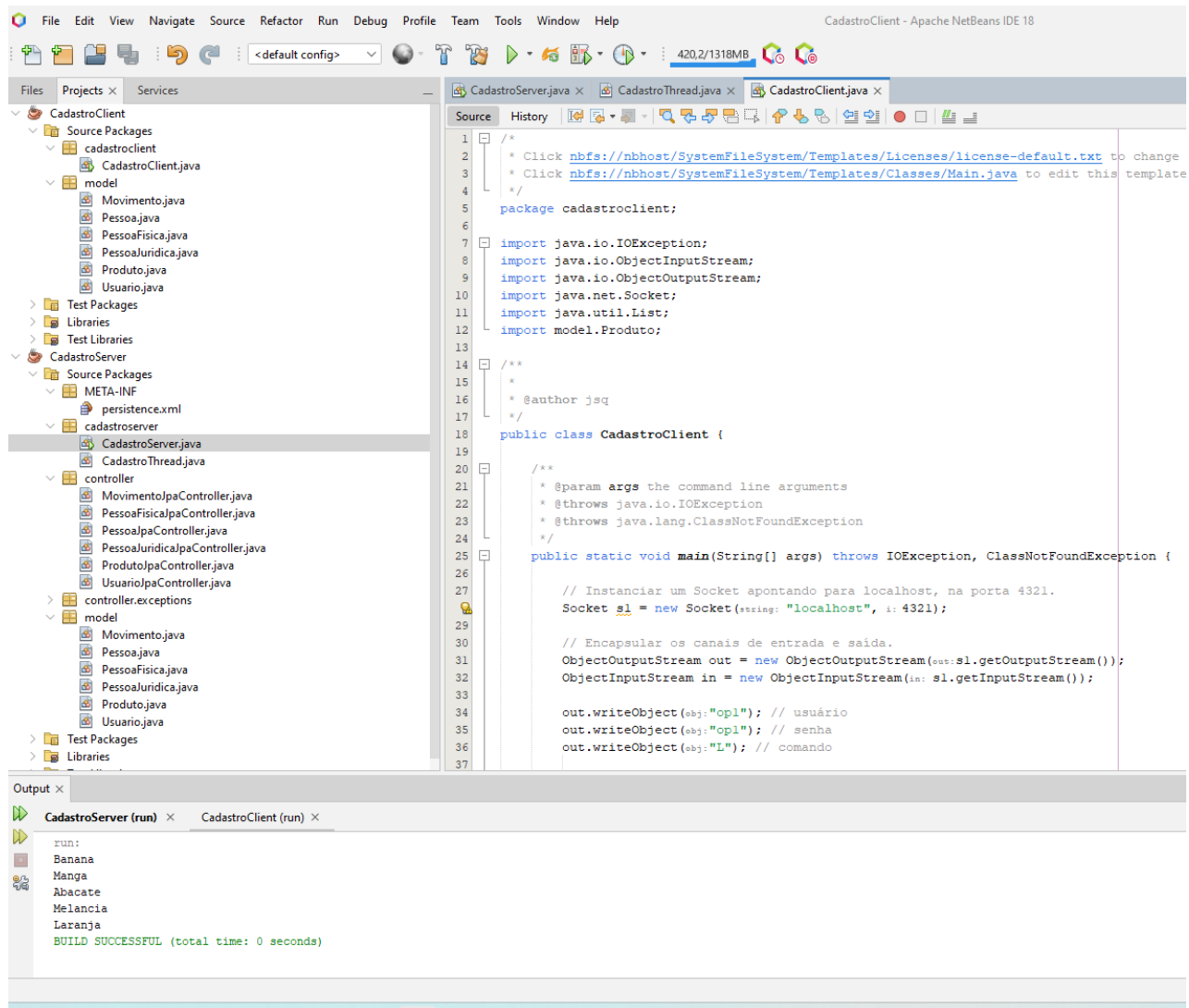
## **3. SOFTWARE UTILIZADO**

- 3.1.** SQL Server Management Studio 19
- 3.2.** Microsoft SQL Server Express 2022
- 3.3.** NetBeans 18
- 3.4.** JDK 8

## **4. PROCEDIMENTOS**

### **4.1. Criando o Servidor e Cliente de Teste**

Nesta etapa, foram criadas duas aplicações, uma cliente e outra servidor. Essas aplicações usam sockets para se comunicar e exibir uma relação dos produtos que estão no banco de dados pelo lado do cliente. Além disso, as aplicações autenticam o usuário usando threads. O objetivo desta etapa foi desenvolver uma aplicação cliente-servidor que permita ao usuário visualizar uma relação dos produtos que estão no banco de dados.



```
1  /*
2  * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change
3  * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
4  */
5  package cadastroclient;
6
7  import java.io.IOException;
8  import java.io.ObjectInputStream;
9  import java.io.ObjectOutputStream;
10 import java.net.Socket;
11 import java.util.List;
12 import model.Produto;
13
14 /**
15 *
16 * @author jsq
17 */
18 public class CadastroClient {
19
20     /**
21     * @param args the command line arguments
22     * @throws java.io.IOException
23     * @throws java.lang.ClassNotFoundException
24     */
25     public static void main(String[] args) throws IOException, ClassNotFoundException {
26
27         // Instanciar um Socket apontando para localhost, na porta 4321.
28         Socket s1 = new Socket("localhost", 4321);
29
30         // Encapsular os canais de entrada e saída.
31         ObjectOutputStream out = new ObjectOutputStream(s1.getOutputStream());
32         ObjectInputStream in = new ObjectInputStream(s1.getInputStream());
33
34         out.writeObject(obj:"opl"); // usuário
35         out.writeObject(obj:"opl"); // senha
36         out.writeObject(obj:"L"); // comando
37     }
38 }
```

run:  
Banana  
Manga  
Abacate  
Melancia  
Laranja  
BUILD SUCCESSFUL (total time: 0 seconds)

#### 4.1.1 Análise e Conclusão:

##### a) Como funcionam as classes Socket e ServerSocket?

São classes do Java que servem para comunicação entre computadores, a classe Socket é usado pelo cliente para se conectar a um servidor e a classe ServerSocket pelo Servidor para aguardar conexões de clientes.

- b) Qual a importância das portas para a conexão com servidores?

As portas, de um computador, utilizam protocolos específicos de comunicação então para haver uma conexão entre dois equipamentos é necessário usar uma de suas portas se atentando para os protocolos que essa porta usa para enviar e receber informação.

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

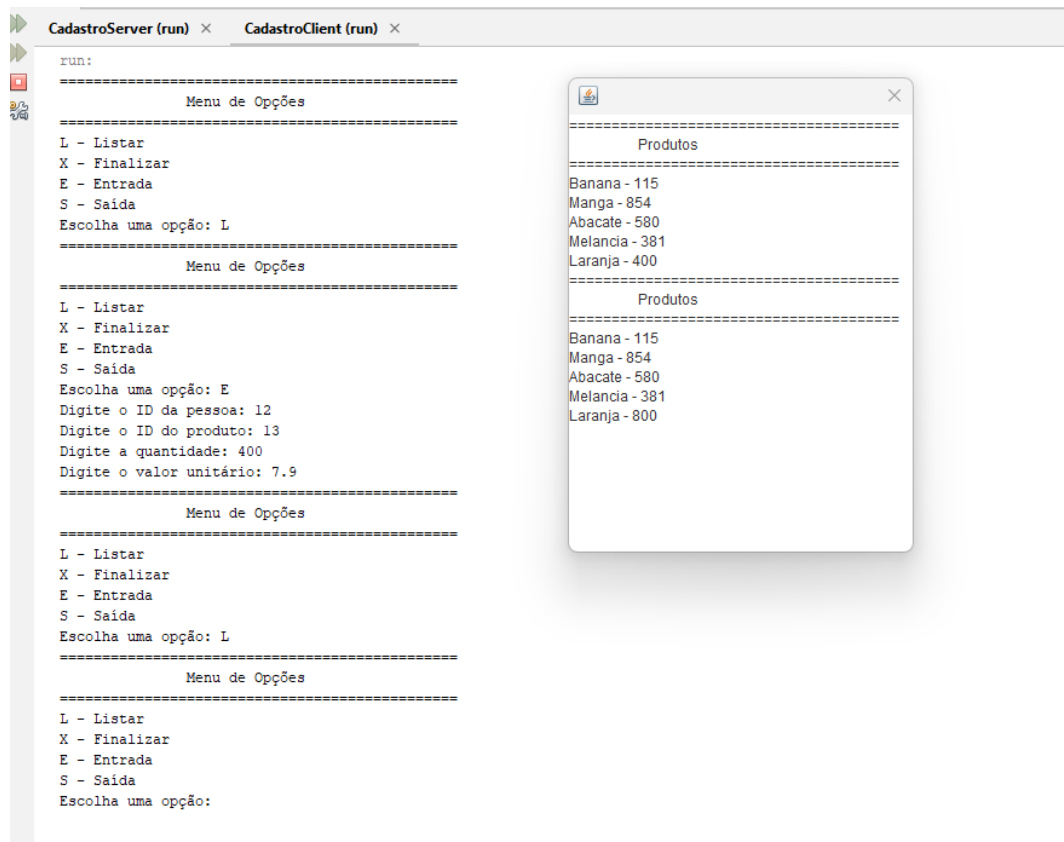
São classes usadas pelo Java para serializar e desserializar objetos. O uso da serialização é necessário para garantir a transmissão de objetos entre aplicativos porque é o único método garantido para transmitir objetos através de qualquer meio, incluindo a rede, arquivos e sockets.

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

Porque só foi passado as classes modelos, o restante das configurações necessárias para efetuar a conexão e manipulação de dados com o banco de dados só foram configurados no servidor.

#### **4.2. Servidor Completo e Cliente Assíncrono**

Nesta fase do desenvolvimento, foram realizadas algumas alterações cruciais na classe do servidor a fim de acomodar funcionalidades adicionais, como a movimentação de produtos. Além disso, foram implementadas duas componentes essenciais do lado do cliente: uma classe de thread e uma interface gráfica (Frame). O resultado dessas adaptações pode ser visualizado a seguir.



#### 4.2.1 Análise e Conclusão:

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

As threads são bastantes uteis para um programa cliente-servidor onde se deseja manter a interface do usuário responsiva enquanto aguarda a chegada de dados do servidor. Comunicação, Pool e Callbacks são maneiras comuns de se utilizar as threads para tratamento assíncrono de respostas do servidor.

- b) Para que serve o método `invokeLater`, da classe `SwingUtilities`?

Garantir que as atualizações da interface gráfica do usuário (GUI) sejam realizadas de forma segura e na thread apropriada, evitando problemas de concorrência e bloqueios na interface do usuário.

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

Através das classes de entrada e saída `ObjectInputStream` e `ObjectOutputStream` utilizando a serialização e a desserialização de objetos.

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

O comportamento síncrono é mais simples de implementar e depurar, mas pode afetar a responsividade. O comportamento assíncrono permite que o cliente continue a execução sem bloquear, mas pode introduzir complexidade adicional e requer gerenciamento cuidadoso de recursos e exceções.

## 5. CONCLUSÃO

Nesta prática, exploramos a criação de aplicações cliente-servidor, tirando proveito de threads e sockets para habilitar o paralelismo entre as aplicações. Além disso, adotamos o padrão de desenvolvimento Model-View-Controller (MVC) para estruturar nossos projetos de forma organizada. Todos os procedimentos e códigos utilizados nesta prática foram documentados e estão disponíveis no repositório GitHub.