Campus: Polo n° 3366 Centro - Garopaba - SC
Curso: Tecnólogo em Desenvolvimento Full Stack
Disciplina: Por que não paralelizar?
Turma: 9001
Nome do Aluno: Luiz Evaldo Pereira
Endereço:

1. **Título**

Implementação de Comunicação Cliente-Servidor Assíncrona com Controle de Movimentações no Estoque em Java

2. **Objetivo**

- Ampliar a comunicação cliente-servidor com suporte a entrada (E) e saída (S) de produtos.
- Desenvolver cliente assíncrono com interface para visualização de mensagens.
- Exercitar uso de threads para comunicação concorrente e atualização GUI com invokeLater.
- Consolidar uso de JPA para persistência de dados no banco SQL Server.

3. **Códigos**

3.1 ThreadModificada.java

```
package cadastroserver;

import controller.*;
import model.*;
import java.net.Socket;
import java.io.*;
import java.util.List;

public class CadastroThreadV2 extends Thread {
    private ProdutoJpaController ctrlProd;
    private UsuarioJpaController ctrlUsu;
    private MovimentoJpaController ctrlMov;
    private PessoaJpaController ctrlPessoa;
    private Socket s1;

    private Usuario usuarioLogado;

    public CadastroThreadV2(ProdutoJpaController ctrlProd,
                            UsuarioJpaController ctrlUsu,
                            MovimentoJpaController ctrlMov,
                            PessoaJpaController ctrlPessoa,
                            Socket s1) {
        this.ctrlProd = ctrlProd;
        this.ctrlUsu = ctrlUsu;
```

```
        this.ctrlMov = ctrlMov;
        this.ctrlPessoa = ctrlPessoa;
        this.s1 = s1;
    }

    @Override
    public void run() {
        try (ObjectOutputStream out = new ObjectOutputStream(s1.getOutputStream());
             ObjectInputStream in = new ObjectInputStream(s1.getInputStream())) {

            // Recebe login e senha
            String login = (String) in.readObject();
            String senha = (String) in.readObject();

            usuarioLogado = ctrlUsu.findUsuario(login, senha);
            if (usuarioLogado == null) {
                // Usuário inválido, fecha conexão
                s1.close();
                return;
            }

            // Loop principal de comandos
            while (true) {
                String comando = (String) in.readObject();

                if ("L".equalsIgnoreCase(comando)) {
                    // Enviar lista de produtos
                    List<Produto> produtos = ctrlProd.findProdutoEntities();
                    out.writeObject(produtos);
                    out.flush();

                                        } else if ("E".equalsIgnoreCase(comando) ||
"S".equalsIgnoreCase(comando)) {
                    // Movimento de Entrada (E) ou Saída (S)

                    // Recebe e configura Movimento
                    Movimento mov = new Movimento();
                    mov.setUsuario(usuarioLogado);
                    mov.setTipo(comando);

                    // Recebe Id Pessoa e atribui
                    Integer idPessoa = (Integer) in.readObject();
                    Pessoa pessoa = ctrlPessoa.findPessoa(idPessoa);
                    if (pessoa == null) {
                        out.writeObject("Pessoa não encontrada!");
                        out.flush();
                        continue; // Pula iteração
                    }
```

```
            mov.setPessoa(pessoa);

            // Recebe Id Produto e atribui
            Integer idProduto = (Integer) in.readObject();
            Produto produto = ctrlProd.findProduto(idProduto);
            if (produto == null) {
                out.writeObject("Produto não encontrado!");
                out.flush();
                continue;
            }
            mov.setProduto(produto);

            // Recebe quantidade e valor unitário
            Integer quantidade = (Integer) in.readObject();
            Double valorUnitario = (Double) in.readObject();
            mov.setQuantidade(quantidade);
            mov.setValorUnitario(valorUnitario);

            // Persiste o movimento
            ctrlMov.create(mov);

            // Atualiza quantidade de produto
            int qtAtual = produto.getQuantidade();
            if ("E".equalsIgnoreCase(comando)) {
                produto.setQuantidade(qtAtual + quantidade);
            } else if ("S".equalsIgnoreCase(comando)) {
                produto.setQuantidade(qtAtual - quantidade);
            }

            ctrlProd.edit(produto);

            out.writeObject("Movimento registrado com sucesso!");
            out.flush();

        } else if ("X".equalsIgnoreCase(comando)) {
            // Finaliza conexão
            break;

        } else {
            out.writeObject("Comando inválido!");
            out.flush();
        }
    }

    s1.close();

} catch (Exception e) {
    e.printStackTrace();
```

```
            }
        }
    }
```

3.2) ClassePrincipalModificada.java

```
package cadastroserver;

import controller.*;
import javax.persistence.*;
import java.net.ServerSocket;
import java.net.Socket;

public class CadastroServer {
    public static void main(String[] args) {
        try {
                                                    EntityManagerFactory emf =
Persistence.createEntityManagerFactory("CadastroPU");

            ProdutoJpaController ctrlProd = new ProdutoJpaController(emf);
            UsuarioJpaController ctrlUsu = new UsuarioJpaController(emf);
            MovimentoJpaController ctrlMov = new MovimentoJpaController(emf);
            PessoaJpaController ctrlPessoa = new PessoaJpaController(emf);

            ServerSocket server = new ServerSocket(4321);
            System.out.println("Servidor aguardando conexões na porta 4321...");

            while (true) {
                Socket clienteSocket = server.accept();
                System.out.println("Cliente conectado: " + clienteSocket.getInetAddress());

                // Instancia nova Thread versão 2
                    CadastroThreadV2 thread = new CadastroThreadV2(ctrlProd, ctrlUsu, ctrlMov,
ctrlPessoa, clienteSocket);
                thread.start();
            }

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

3.3) CadastroClientV2

```
package cadastroclientv2;

import model.Produto;
import javax.swing.*;
```

```
import java.io.*;
import java.net.Socket;
import java.util.List;

public class CadastroClientV2 {

    public static void main(String[] args) {
        try (Socket socket = new Socket("localhost", 4321);
             ObjectOutputStream out = new ObjectOutputStream(socket.getOutputStream());
             ObjectInputStream in = new ObjectInputStream(socket.getInputStream());
                                        BufferedReader teclado = new BufferedReader(new
InputStreamReader(System.in))) {

            // Login e senha (exemplo op1/op1)
            out.writeObject("op1");
            out.writeObject("op1");
            out.flush();

            // Cria GUI para saída de mensagens
            SaidaFrame saida = new SaidaFrame();
            saida.setVisible(true);

            // Inicia thread assíncrona que lê servidor continuamente
            ThreadClient tClient = new ThreadClient(in, saida.texto);
            tClient.start();

            while (true) {
                  System.out.println("\nMenu:\nL - Listar Produtos\nE - Entrada de Produto\nS -
Saída de Produto\nX - Sair");
                System.out.print("Comando: ");
                String cmd = teclado.readLine();

                if (cmd == null) continue;
                cmd = cmd.trim().toUpperCase();

                if (cmd.equals("X")) {
                    out.writeObject("X");
                    out.flush();
                    break;
                }

                if (cmd.equals("L")) {
                    out.writeObject("L");
                    out.flush();

                } else if (cmd.equals("E") || cmd.equals("S")) {
                    out.writeObject(cmd);
                    out.flush();
```

```
                System.out.print("Id da Pessoa: ");
                int idPessoa = Integer.parseInt(teclado.readLine());
                out.writeObject(idPessoa);
                out.flush();

                System.out.print("Id do Produto: ");
                int idProduto = Integer.parseInt(teclado.readLine());
                out.writeObject(idProduto);
                out.flush();

                System.out.print("Quantidade: ");
                int qtd = Integer.parseInt(teclado.readLine());
                out.writeObject(qtd);
                out.flush();

                System.out.print("Valor Unitário: ");
                double valU = Double.parseDouble(teclado.readLine());
                out.writeObject(valU);
                out.flush();

            } else {
                System.out.println("Comando inválido!");
            }
        }

    } catch (Exception e) {
        e.printStackTrace();
    }
  }
}
```

3.4)SaidaFrame.java

```
package cadastroclientv2;

import javax.swing.*;

public class SaidaFrame extends JDialog {
   public JTextArea texto;

   public SaidaFrame() {
      setTitle("Saídas do Servidor");
      setBounds(100, 100, 400, 300);
      setModal(false);
      setDefaultCloseOperation(JDialog.DISPOSE_ON_CLOSE);
      texto = new JTextArea();
      texto.setEditable(false);
      JScrollPane scroll = new JScrollPane(texto);
```

```
        add(scroll);
    }
}
```

3.5) ThreadClient

```
package cadastroclientv2;

import model.Produto;
import javax.swing.*;
import java.io.ObjectInputStream;
import java.util.List;

public class ThreadClient extends Thread {
    private ObjectInputStream entrada;
    private JTextArea textArea;

    public ThreadClient(ObjectInputStream entrada, JTextArea textArea) {
        this.entrada = entrada;
        this.textArea = textArea;
    }

    @Override
    public void run() {
        try {
            while (true) {
                Object obj = entrada.readObject();

                if (obj instanceof String) {
                    String msg = (String) obj;
                    SwingUtilities.invokeLater(() -> {
                        textArea.append(msg + "\n");
                    });
                } else if (obj instanceof List<?>) {
                    List<Produto> produtos = (List<Produto>) obj;

                    SwingUtilities.invokeLater(() -> {
                        textArea.append("Lista de Produtos (Nome - Quantidade):\n");
                        for (Produto p : produtos) {
                                        textArea.append(String.format("- %s : %d\n", p.getNome(),
p.getQuantidade()));
                        }
                    });
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

4. **Análise e conclusão:**

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

Threads permitem que o cliente aguarde e receba mensagens do servidor de forma independente do fluxo principal do programa, evitando bloqueios (como bloqueio na leitura da rede), permitindo que o usuário interaja com o sistema enquanto a comunicação acontece em segundo plano.

Para que serve o método invokeLater, da classe SwingUtilities?

O método `invokeLater` agenda uma tarefa para rodar na Event Dispatch Thread (EDT) do Swing, que é a thread responsável pela manipulação da interface gráfica. Isso garante que atualizações em componentes Swing sejam feitas de forma segura, evitando problemas de concorrência.

Como os objetos são enviados e recebidos pelo Socket Java?

Objetos são transmitidos por meio de streams de objetos (`ObjectOutputStream` e `ObjectInputStream`). Os objetos devem implementar a interface `Serializable` para serem convertidos em bytes para envio e reconstruídos na outra ponta da conexão.

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

Síncrono: o cliente bloqueia enquanto espera resposta do servidor, congelando interface ou fluxo, reduzindo a responsividade.

Assíncrono: o cliente usa threads separadas para ler dados da rede, permitindo que a interface fique responsiva, e múltiplas operações possam ocorrer simultaneamente.