



# Universidade Estácio de Sá

---

- Curso: Desenvolvimento Full Stack
  - Disciplina: RPG0018 - Por que não paralelizar
  - Semestre Letivo: 2025.1
  - Turma: 9001
  - Campus: Parangaba
  - Aluno: Caio Viana Castelo Branco
  - Matrícula: 202307465925
- 

## Missão Prática | Nível 5 | Mundo 3

---

**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.**

**Procedimento 1: Criando o Servidor e Cliente de Teste**

**Procedimento 2: Servidor Completo e Cliente Assíncrono**

---

## Objetivos da Prática

- Criar servidores Java com base em Sockets.
- Criar clientes síncronos para servidores com base em Sockets.
- Criar clientes assíncronos para servidores com base em Sockets.
- Utilizar Threads para implementação de processos paralelos.
- 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.

---

## Códigos

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

- CadastroClient.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to view the code.
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit the template
 */
package cadastroclient;

import java.io.IOException;
import java.io.InputStreamReader;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.PrintStream;
import java.net.Socket;
import java.util.List;
import java.util.Scanner;
import model.Produto;

/**
 *
 * @author caio
 */
public class CadastroClient {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws ClassNotFoundException,
        SocketException, IOException {
        Socket socket = new Socket("localhost", 4321);
        ObjectOutputStream out = new ObjectOutputStream(socket.getOutputStream());
        ObjectInputStream in = new ObjectInputStream(socket.getInputStream());

        out.writeObject("op1");
    }
}
```

```

        out.writeObject("op1");

        System.out.println((String) in.readObject());

        out.writeObject("L");

        List<Produto> produtos = (List<Produto>) in.readObject();
        for (Produto produto : produtos) {
            System.out.println(produto.getNome());
        }

        out.close();
        in.close();
        socket.close();
    }
}

```

- CadastroServer.java

```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to view the code
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit the template
 */
package cadastrserver;

import controller.ProdutoJpaController;
import controller.UsuarioJpaController;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.net.ServerSocket;
import java.net.Socket;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

/**
 *
 * @author caio
 */
public class CadastroServer {

```

```

/**
 * @param args the command line arguments
 */
public static void main(String[] args) throws IOException{
    ServerSocket serverSocket = new ServerSocket(4321);
    EntityManagerFactory emf = Persistence.createEntityManagerFactory
    ProdutoJpaController ctrl = new ProdutoJpaController(emf);
    UsuarioJpaController ctrlUsu = new UsuarioJpaController(emf);

    while (true) {

        Socket clienteSocket = serverSocket.accept();
        System.out.println("Cliente conectado: " + clienteSocket.getI

        CadastroThread thread = new CadastroThread(ctrl, ctrlUsu, cli

        thread.start();
        System.out.println("Aguardando nova conexÃo...");
    }

}
}

```

- CadastroThread.java

```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to e
 */
package cadastrserver;

import controller.ProdutoJpaController;
import controller.UsuarioJpaController;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.util.Scanner;
import model.Usuario;

/**

```

```

*
* @author caio
*/
public class CadastroThread extends Thread {

    private ProdutoJpaController ctrl;
    private UsuarioJpaController ctrlUsu;
    private Socket s1;
    private ObjectOutputStream out;
    private ObjectInputStream in;

    CadastroThread (ProdutoJpaController ctrl, UsuarioJpaController ctrl
    this.ctrl = ctrl;
        this.ctrlUsu = ctrlUsu;
        this.s1 = s1;
    }

    @Override
    public void run(){

        String login = "";

        try{
            out = new ObjectOutputStream(s1.getOutputStream());
            in = new ObjectInputStream(s1.getInputStream());

            System.out.println("Cliente conectado.");

            login = (String) in.readObject();
            String senha = (String) in.readObject();

            Usuario usuario = ctrlUsu.findUsuario(login, senha);
            if (usuario == null) {
                System.out.println("Usuário inválido."); //Login="+ log
                out.writeObject("Usuário inválido.");
                return;
            }

            System.out.println("Usuário conectado.");
            out.writeObject("Usuário conectado.");

            System.out.println("Aguardando comandos...");

```

```

        String comando = (String) in.readObject();

        if (comando.equals("L")) {
            System.out.println("Listando produtos.");
            out.writeObject(ctrl.findProdutoEntities());
        }
    } catch (IOException | ClassNotFoundException e) {
        e.printStackTrace();
    } finally {
        close();
        System.out.println("Conexão finalizada.");
    }

}

private void close() {
    try {
        if (out != null) {
            out.close();
        }
        if (in != null) {
            in.close();
        }
        if (s1 != null) {
            s1.close();
        }
    } catch (IOException ex) {
        System.out.println("Falha ao finalizar conexão.");
    }
}
}
}

```

## Procedimento 2: Servidor Completo e Cliente Assíncrono

- CadastroClientv2.java

```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to view the code.
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit the template.
 */
package cadastroclient;

import java.io.BufferedReader;
import java.io.IOException;

```

```

import java.io.InputStreamReader;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.PrintStream;
import java.net.Socket;
import java.util.List;
import java.util.Scanner;
import model.Produto;

/**
 *
 * @author caio
 */

public class CadastroClientv2 {

    private static ObjectOutputStream socketOut;
    private static ObjectInputStream socketIn;
    private static ThreadClient threadClient;

    /**
     * @param args the command line arguments
     */

    public static void main(String[] args) throws ClassNotFoundException,
        Socket socket = new Socket("localhost", 4321);
        socketOut = new ObjectOutputStream(socket.getOutputStream());
        socketIn = new ObjectInputStream(socket.getInputStream());

        BufferedReader reader = new BufferedReader(new InputStreamReader(

        SaidaFrame saidaFrame = new SaidaFrame();
        saidaFrame.setVisible(true);

        threadClient = new ThreadClient(socketIn, saidaFrame.texto);
        threadClient.start();

        socketOut.writeObject("op1");

        socketOut.writeObject("op1");

```

```

Character comando = ' ';
try {
    while (!comando.equals('X')) {
        System.out.println("Escolha uma opção:");
        System.out.println("L - Listar | X - Finalizar | E - Entr

        comando = reader.readLine().charAt(0);

        processaComando(reader, comando);
    }
} catch(Exception e) {
    e.printStackTrace();
} finally {
    saidaFrame.dispose();
    socketOut.close();
    socketIn.close();
    socket.close();
    reader.close();
}
}

static void processaComando(BufferedReader reader, Character comando
    socketOut.writeChar(comando);
    socketOut.flush();

    switch (comando) {
        case 'L':
            break;
        case 'S':
        case 'E':
            socketOut.flush();

            System.out.println("Digite o Id da pessoa:");
            int idPessoa = Integer.parseInt(reader.readLine());
            System.out.println("Digite o Id do produto:");
            int idProduto = Integer.parseInt(reader.readLine());
            System.out.println("Digite a quantidade:");
            int quantidade = Integer.parseInt(reader.readLine());
            System.out.println("Digite o valor unitário:");
            long valorUnitario = Long.parseLong(reader.readLine());

            socketOut.writeInt(idPessoa);
            socketOut.flush();
            socketOut.writeInt(idProduto);

```



```

        socketOut.flush();
        socketOut.writeInt(quantidade);
        socketOut.flush();
        socketOut.writeLong(valorUnitario);
        socketOut.flush();
        break;
    case 'X':
        threadClient.cancela();
        break;
    default:
        System.out.println("Opção inválida!");
    }
}
}
}

```

- SaidaFrame

```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to view and
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit and
 */
package cadastroclient;
import javax.swing.*.*;
/**
 *
 * @author caio
 */
public class SaidaFrame extends JDialog {
    public JTextArea texto;

    public SaidaFrame() {
        setBounds(100, 100, 400, 300);

        setModal(false);

        texto = new JTextArea(25, 40);
        texto.setEditable(false); // Bloqueia edição do campo de texto

        JScrollPane scroll = new JScrollPane(texto);
        scroll.setHorizontalScrollBarPolicy(JScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);
        add(scroll);
    }
}

```

```
}  
}
```

- ThreadClient.java

```
/*  
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to view the code.  
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to view the code.  
 */  
package cadastroclient;  
import java.io.IOException;  
import java.io.ObjectInputStream;  
import java.net.SocketException;  
import java.util.List;  
import javax.swing.JTextArea;  
import javax.swing.SwingUtilities;  
import model.Produto;  
  
/**  
 *  
 * @author caio  
 */  
public class ThreadClient extends Thread {  
    private ObjectInputStream entrada;  
    private JTextArea textArea;  
    private Boolean cancelada;  
  
    public ThreadClient(ObjectInputStream entrada, JTextArea textArea) {  
        this.entrada = entrada;  
        this.textArea = textArea;  
        this.cancelada = false;  
    }  
  
    @Override  
    public void run() {  
        while (!cancelada) {  
            try {  
                Object resposta = entrada.readObject();  
                SwingUtilities.invokeLater(() -> {  
                    processaResposta(resposta);  
                });  
            } catch (IOException | ClassNotFoundException e) {  
                if (!cancelada) {  

```

```

        System.err.println(e);
    }
}

}

}

public void cancela() {
    cancelada = true;
}

private void processaResposta(Object resposta) {
    textArea.append(">> Nova comunicaÃ§Ã£o em " + java.time.LocalDateTime.now().format(DateTimeFormatter.ofPattern("dd/MM/yyyy HH:mm:ss")) + "\n");

    if (resposta instanceof String) {
        textArea.append((String) resposta + "\n");
    } else if (resposta instanceof List<?>) {
        textArea.append("> Listagem dos produtos:\n");
        List<Produto> lista = (List<Produto>) resposta;
        for (Produto item : lista) {
            textArea.append("Produto=[" + item.getNome() + "], Quantidade=" + item.getQuantidade() + "\n");
        }
    }
    textArea.append("\n");
    textArea.setCaretPosition(textArea.getDocument().getLength());
}
}

```

- CadastroServer.java

```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to view and
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit the template
 */
package cadastroserver;

import controller.MovimentoJpaController;
import controller.PessoaJpaController;
import controller.ProdutoJpaController;
import controller.UsuarioJpaController;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintStream;
import java.net.ServerSocket;

```

```

import java.net.Socket;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

/**
 *
 * @author caio
 */
public class CadastroServer {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) throws IOException{
        ServerSocket serverSocket = new ServerSocket(4321);
        EntityManagerFactory emf = Persistence.createEntityManagerFactory(
            "ProdutoJpaController");
        ProdutoJpaController ctrl = new ProdutoJpaController(emf);
        UsuarioJpaController ctrlUsu = new UsuarioJpaController(emf);
        MovimentoJpaController ctrlMov = new MovimentoJpaController(emf);
        PessoaJpaController ctrlPessoa = new PessoaJpaController(emf);

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

            CadastroThreadv2 thread = new CadastroThreadv2(ctrl, ctrlUsu,
                ctrlMov, ctrlPessoa);

            thread.start();
            System.out.println("Aguardando nova conexão...");
        }
    }
}

```

- CadastroThreadv2.java

```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to
 * view the source code.
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to
 * edit the template.
 */
package cadastroserver;

```

```

import controller.MovimentoJpaController;
import controller.PessoaJpaController;
import controller.ProdutoJpaController;
import controller.UsuarioJpaController;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.util.Scanner;
import java.util.logging.Level;
import java.util.logging.Logger;
import model.Movimento;
import model.Produto;
import model.Usuario;

/**
 *
 * @author caio
 */
public class CadastroThreadv2 extends Thread {

    private ProdutoJpaController ctrl;
    private UsuarioJpaController ctrlUsu;
    private MovimentoJpaController ctrlMov;
    private PessoaJpaController ctrlPessoa;
    private Socket s1;
    private ObjectOutputStream out;
    private ObjectInputStream in;
    private Usuario usuario;
    private Boolean continuaProcesso = true;

    CadastroThreadv2 (ProdutoJpaController ctrl, UsuarioJpaController ct
    this.ctrl = ctrl;
        this.ctrlUsu = ctrlUsu;
        this.ctrlMov = ctrlMov;
        this.ctrlPessoa = ctrlPessoa;
        this.s1 = s1;
    }

    @Override
    public void run(){

```

```

String login = "";

try{
    out = new ObjectOutputStream(s1.getOutputStream());
    in = new ObjectInputStream(s1.getInputStream());

    System.out.println("Cliente conectado.");

    login = (String) in.readObject();
    String senha = (String) in.readObject();
    usuario = ctrlUsu.findUsuario(login, senha);

    if (usuario == null) {
        System.out.println("Usuário inválido.");
        out.writeObject("Usuário inválido.");
        return;
    }

    System.out.println("Usuário conectado.");
    out.writeObject("Usuário conectado.");
    out.flush();

    while (continuaProcesso) {
        continuaProcesso = processaComando();
    }

}catch (IOException | ClassNotFoundException e) {
    e.printStackTrace();
} catch (Exception ex) {
    Logger.getLogger(CadastroThreadv2.class.getName()).log(Level.
} finally {
    close();
    System.out.println("Conexão finalizada.");
}

}

private Boolean processaComando() throws Exception {
    System.out.println("Aguardando comandos...");
    Character comando = in.readChar();

```

```

switch (comando) {
    case 'L':
        System.out.println("Comando recebido, listando produtos."
            out.writeObject(ctrl.findProdutoEntities());
        continuaProcesso = true;
        return true;
    case 'E':
        continuaProcesso = true;
        return true;
    case 'S':
        System.out.println("Comando Movimento tipo [" + comando + "
            int idPessoa = in.readInt();
            int idProduto = in.readInt();
            int quantidade = in.readInt();
            Float valorUnitario = in.readFloat();

        Produto produto = ctrl.findProduto(idProduto);
        if (produto == null) {
            out.writeObject("Produto inválido.");
            continuaProcesso = true;
            return true;
        }

        if (comando.equals('E')) {
            produto.setQuantidade(produto.getQuantidade() + quant
            continuaProcesso = true;
            return true;
        } else if (comando.equals('S')) {
            produto.setQuantidade(produto.getQuantidade() - quant
            continuaProcesso = true;
            return true;
        }

        ctrl.edit(produto);

        Movimento movimento = new Movimento();
        movimento.setTipo(comando);
        movimento.setUsuarioidUsuario(usuario);
        movimento.setPessoaIdpessoa(ctrlPessoa.findPessoa(idPessoa
        movimento.setProdutoIdproduto(produto);
        movimento.setQuantidade(quantidade);
        movimento.setValorUnitario(valorUnitario);

        ctrlMov.create(movimento);

```

```

        out.writeObject("Movimento registrado com sucesso.");
        out.flush();
        System.out.println("Movimento registrado com sucesso.");
        continuaProcesso = true;
        return true;
    case 'X':
        continuaProcesso = false;
        return false;
    default:
        System.out.println("Opção inválida!");
        continuaProcesso = false;
        return true;
    }
}

private void close() {
    try {
        if (out != null) {
            out.close();
        }
        if (in != null) {
            in.close();
        }
        if (s1 != null) {
            s1.close();
        }
    } catch (IOException ex) {
        System.out.println("Falha ao fechar conexão.");
    }
}
}

```

---

## Análise e Conclusão

---

- Como funcionam as classes Socket e ServerSocket?

É empregado no servidor para aguardar e aceitar conexões vindas da rede. No Socket é utilizado no cliente para estabelecer uma conexão com o servidor.



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

Permitem que o cliente e o servidor se comuniquem entre si criando um canal identificado, evitando conflitos.

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

Possibilitam que objetos sejam serializados, ou seja, convertidos em um formato que pode ser transmitido ou armazenado em arquivos. A serialização é essencial para transmitir objetos pela rede ou persisti-los em arquivos, pois transforma os objetos em bytes que podem ser reconstruídos posteriormente.

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

A lógica de acesso ao banco de dados fica à cargo das classes `Controllers`, que neste caso, existem apenas do lado do Servidor, garantindo assim o isolamento do acesso ao banco de dados.

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

Através das threads no cliente, é possível atualizar os dados na interface, no caso através da classe `SaidaFrame` que herda de `JDialog`, sem que o processo principal seja interrompido e sem que a interface fique bloqueada, permitindo o cliente ficar sempre “ouvindo” as respostas do servidor.

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

É utilizado para agendar a execução de um trecho de código na Thread de eventos Swing, também conhecida como EDT (`Event Dispatch Thread`). Essa técnica é essencial para garantir que as operações relacionadas à interface do usuário sejam realizadas na Thread apropriada, prevenindo problemas de concorrência e assegurando a responsividade da interface do usuário em aplicativos Swing.

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

Para enviar e receber objetos via Socket são utilizadas as classes `ObjectInputStream` e `ObjectOutputStream`. Para enviar um objeto, o método `writeObject()` da classe `ObjectOutputStream` é chamado passando o objeto que como argumento. Para receber um objeto, o método `readObject()` da classe `ObjectInputStream` é chamado. Há outros métodos para envio e recebimento apropriados para cada tipo, por exemplo: `writeChar()`, `writeInt()`, `writeLong()`, `readChar()`, `readInt()`, `readLong()` dentro vários outros.

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

No modelo síncrono, as operações de socket bloqueiam o processo do cliente até sua conclusão, o que significa que o cliente fica parado, aguardando a resposta do servidor antes de continuar com outras tarefas. Em contrapartida, no modelo assíncrono, as operações de socket não bloqueiam o processo do cliente, sendo executadas em segundo plano. Isso permite que o cliente prossiga com outras tarefas enquanto aguarda a conclusão das operações de socket, proporcionando maior responsividade ao aplicativo e evitando atrasos no processamento.