



Desenvolvimento Full Stack

Felipe Marques de Almeida - 202208291929

Polo Rodovia Br 407 - Juazeiro - Ba

RPG0018 - Por que não paralelizar – Turma 22.3 – 3º Semestre

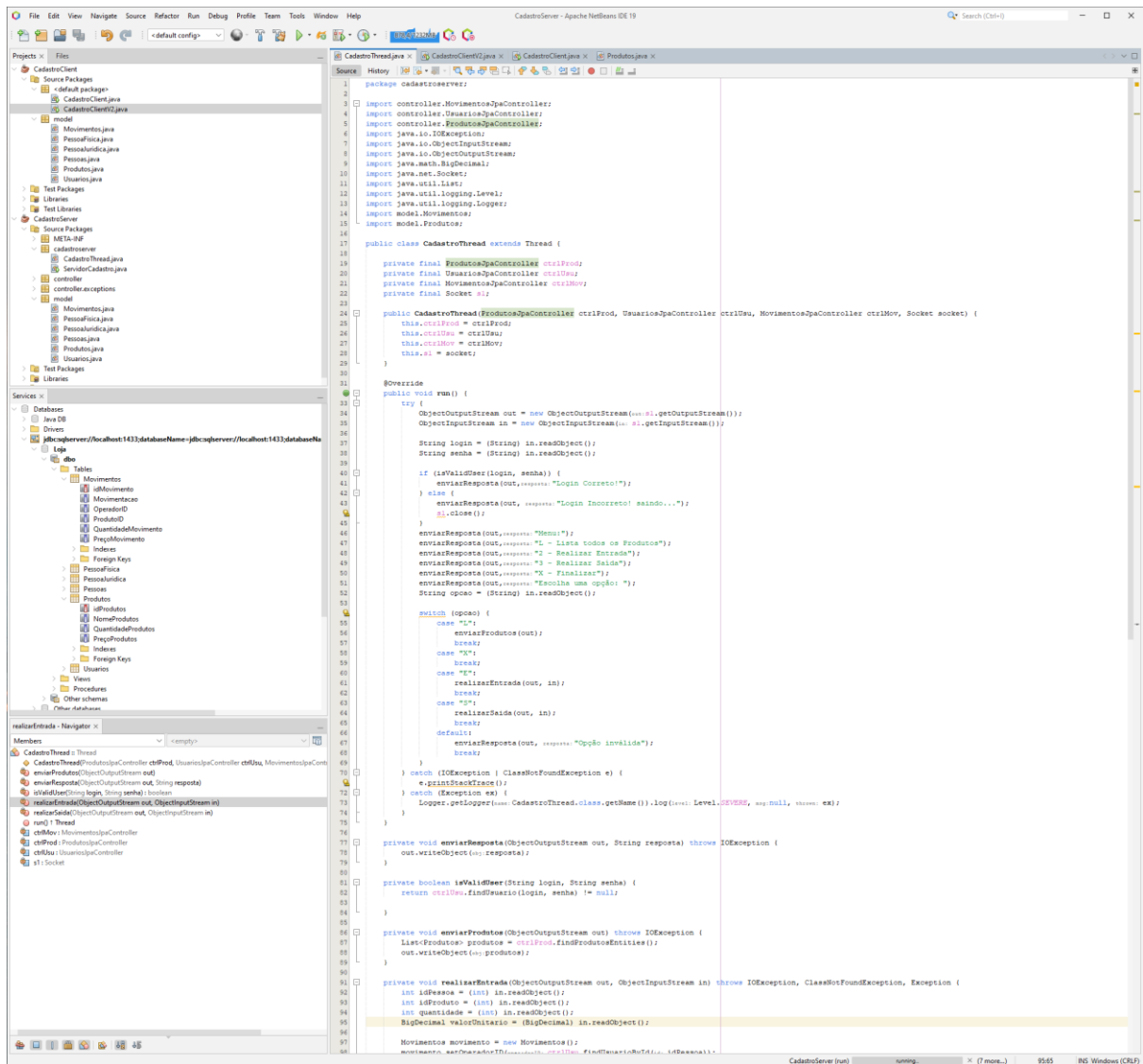
GitHub: [Kifflom2108/Missao-Pratica-Nivel-5-Mundo-3](https://github.com/Kifflom2108/Missao-Pratica-Nivel-5-Mundo-3) (github.com)

Objetivo da Prática

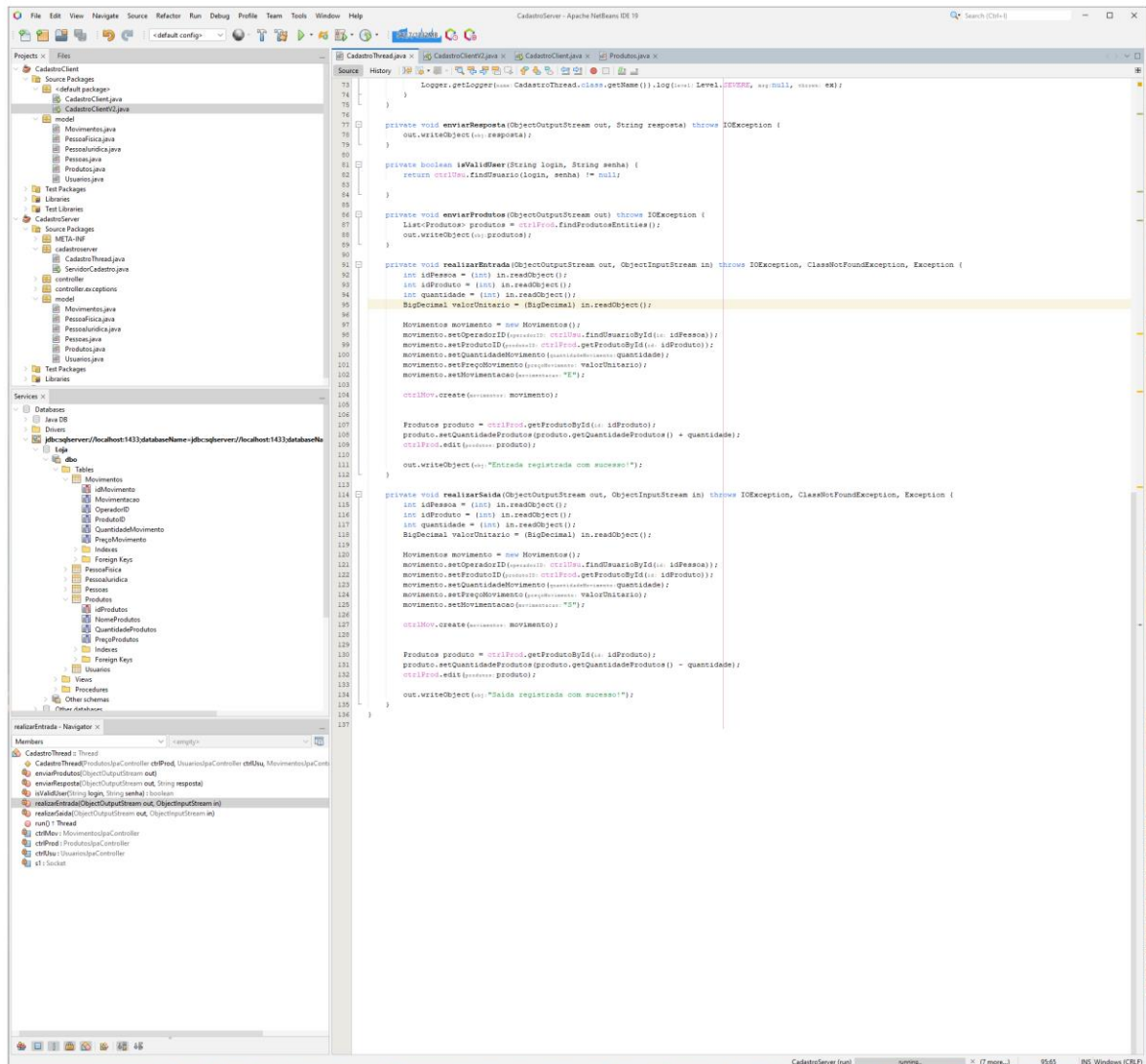
2º Procedimento | Servidor Completo e Cliente Assíncrono

Codigos: (Todas as imagens com qualidade original estão disponível no repositório!)

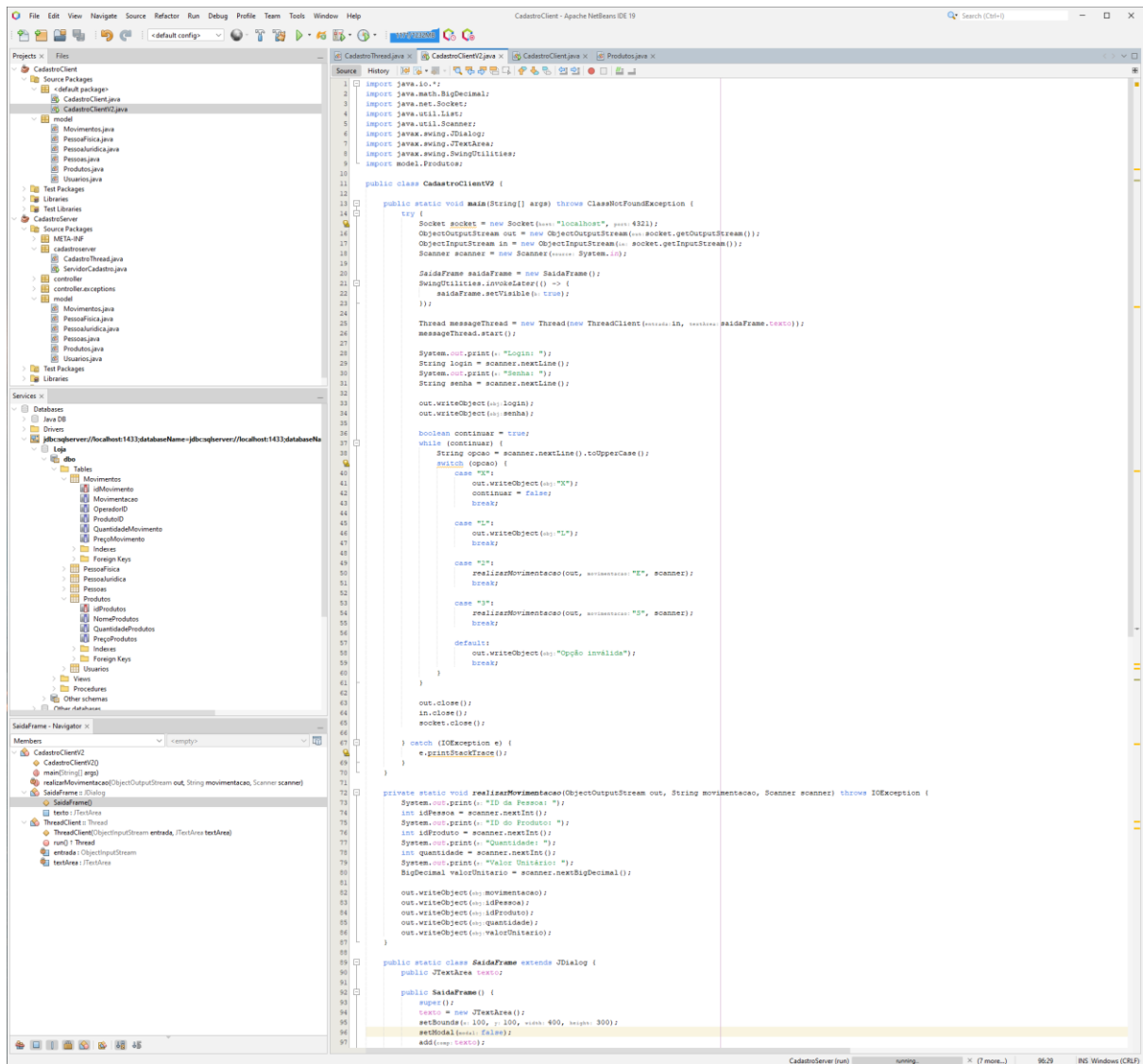
CadastroThread:

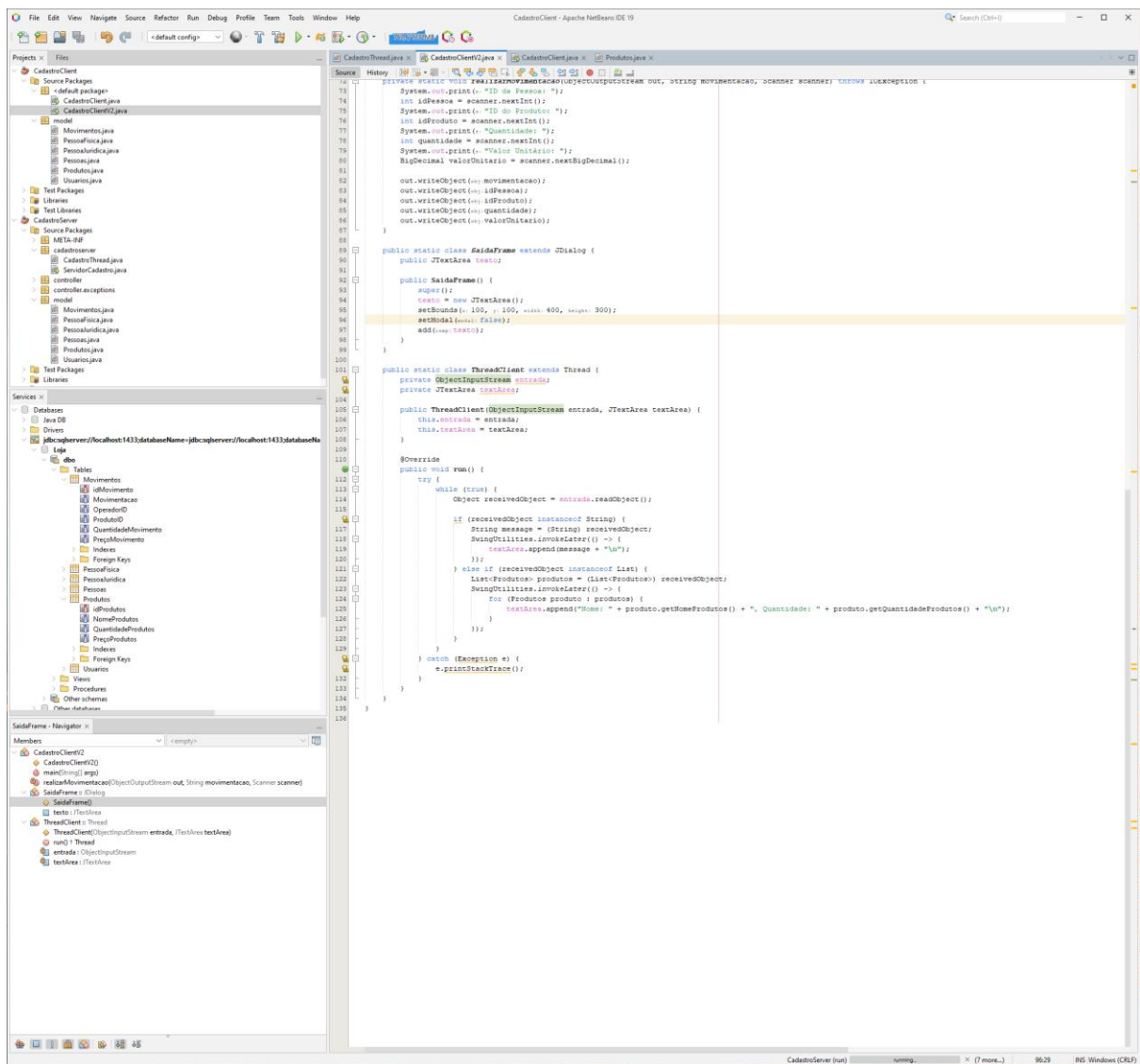


```
1 package cadastroserver;
2
3 import controller.Movimento3paController;
4 import controller.Usuario3paController;
5 import controller.Produto3paController;
6 import java.io.IOException;
7 import java.io.ObjectInputStream;
8 import java.io.ObjectOutputStream;
9 import java.math.BigDecimal;
10 import java.net.Socket;
11 import java.util.List;
12 import java.util.logging.Level;
13 import java.util.logging.Logger;
14 import model.Movimento;
15 import model.Produto;
16
17 public class CadastroThread extends Thread {
18
19     private final Produto3paController ctrlProd;
20     private final Usuario3paController ctrlUsu;
21     private final Movimento3paController ctrlMov;
22     private final Socket s1;
23
24     public CadastroThread(Produto3paController ctrlProd, Usuario3paController ctrlUsu, Movimento3paController ctrlMov, Socket socket) {
25         this.ctrlProd = ctrlProd;
26         this.ctrlUsu = ctrlUsu;
27         this.ctrlMov = ctrlMov;
28         this.s1 = socket;
29     }
30
31     @Override
32     public void run() {
33         try {
34             ObjectOutputStream out = new ObjectOutputStream(s1.getOutputStream());
35             ObjectInputStream in = new ObjectInputStream(s1.getInputStream());
36
37             String login = (String) in.readObject();
38             String senha = (String) in.readObject();
39
40             if (isValidUser(login, senha)) {
41                 enviaResponse(out, response: "Login Correto!");
42             } else {
43                 enviaResponse(out, response: "Login Incorreto! anade...");
44                 s1.close();
45             }
46
47             enviaResponse(out, response: "Menu");
48             enviaResponse(out, response: "1 - Lista todos os Produtos");
49             enviaResponse(out, response: "2 - Realizar Entrada");
50             enviaResponse(out, response: "3 - Realizar Saída");
51             enviaResponse(out, response: "X - Finalizar");
52             enviaResponse(out, response: "Escolha uma opção: ");
53             String opcao = (String) in.readObject();
54
55             switch (opcao) {
56                 case "1":
57                     enviaProdutos(out);
58                     break;
59                 case "2":
60                     realizarEntrada(out, in);
61                     break;
62                 case "3":
63                     realizarSaida(out, in);
64                     break;
65                 default:
66                     enviaResponse(out, response: "Opção inválida");
67                     break;
68             }
69         } catch (IOException | ClassNotFoundException e) {
70             e.printStackTrace();
71         } catch (Exception ex) {
72             Logger.getLogger(CadastroThread.class.getName()).log(Level.SEVERE, null, ex);
73         }
74     }
75
76     private void enviaResponse(ObjectOutputStream out, String response) throws IOException {
77         out.writeObject(response);
78     }
79
80     private boolean isValidUser(String login, String senha) {
81         return ctrlUsu.findUsuario(login, senha) != null;
82     }
83
84     private void enviaProdutos(ObjectOutputStream out) throws IOException {
85         List<Produto> produtos = ctrlProd.findProdutosEntities();
86         out.writeObject(produtos);
87     }
88
89     private void realizarEntrada(ObjectOutputStream out, ObjectInputStream in) throws IOException, ClassNotFoundException, Exception {
90         int idUsuario = (int) in.readObject();
91         int idProduto = (int) in.readObject();
92         int quantidade = (int) in.readObject();
93         BigDecimal valorUnitario = (BigDecimal) in.readObject();
94
95         Movimento movimento = new Movimento();
96         movimento.setUsuario(ctrlUsu.findUsuarioById(idUsuario));
97     }
98 }
```

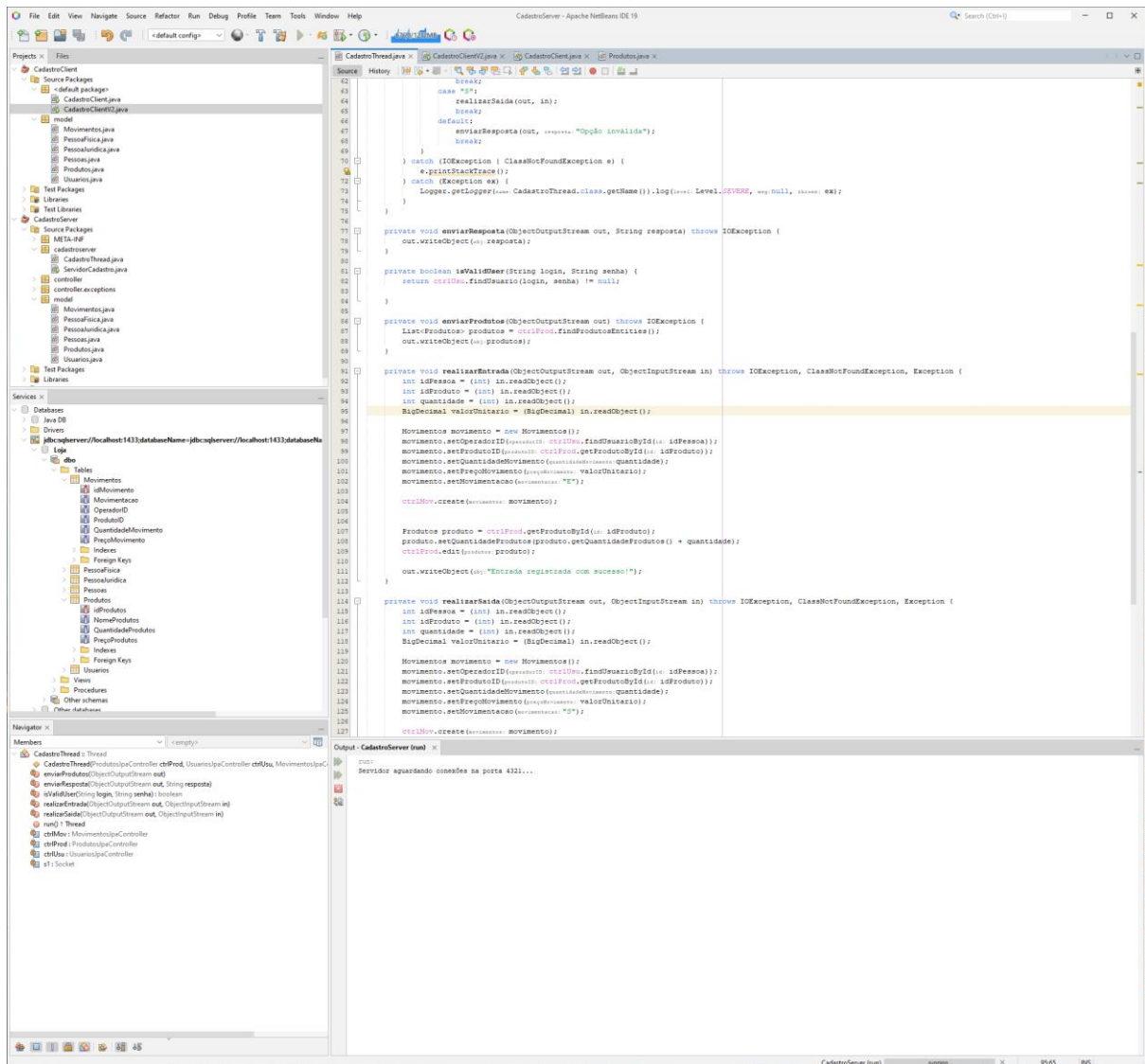


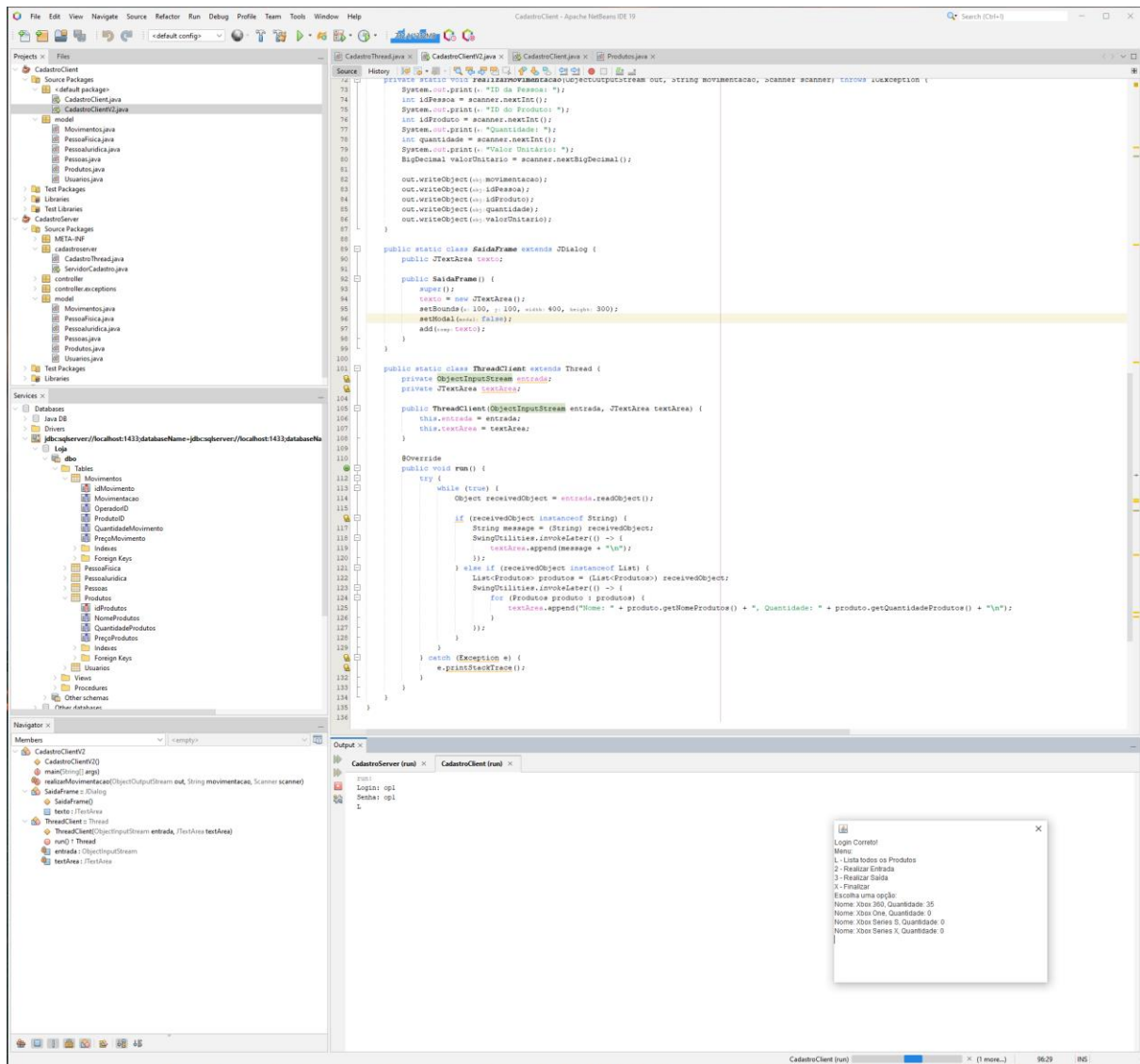
CadastroV2

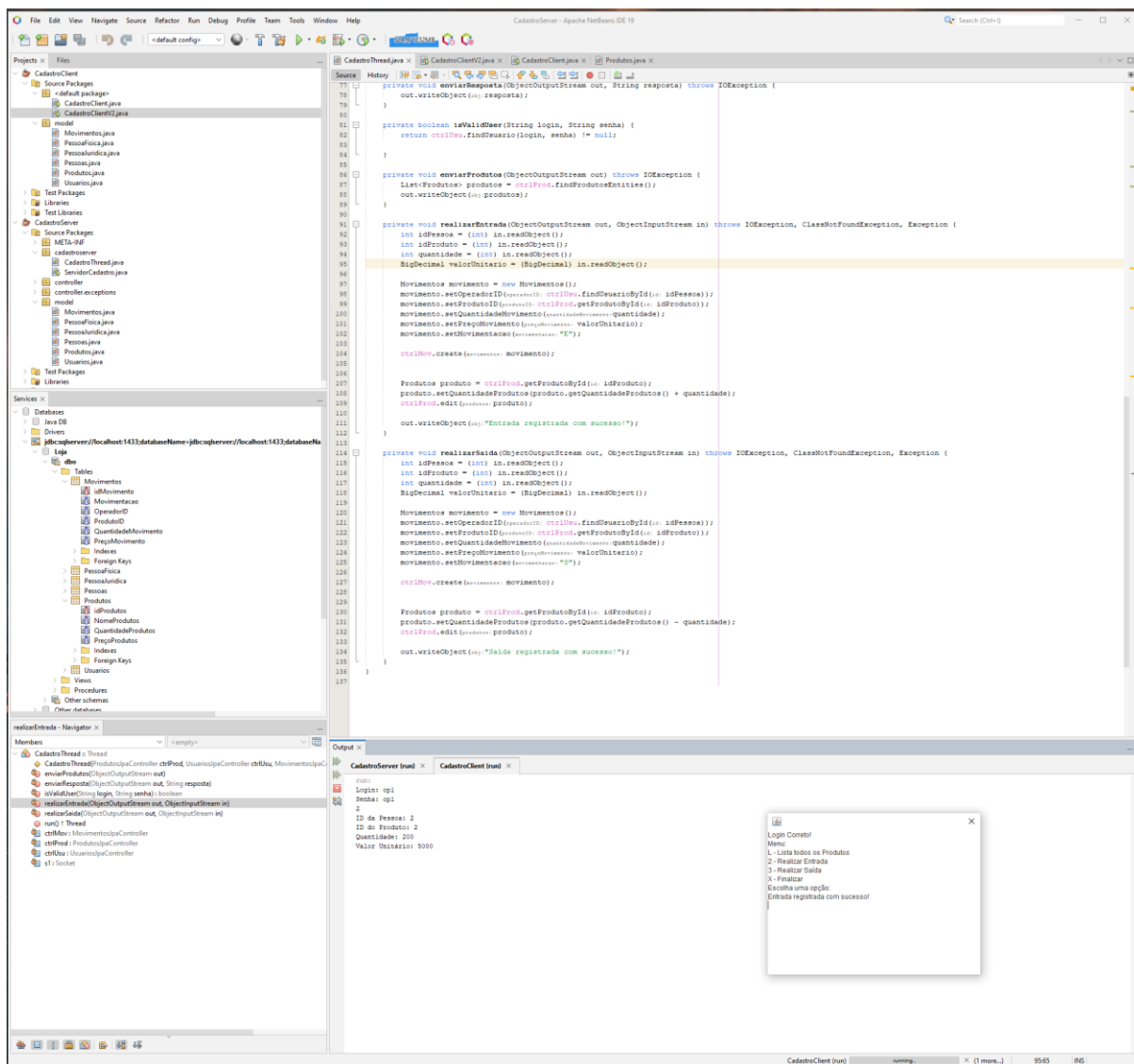


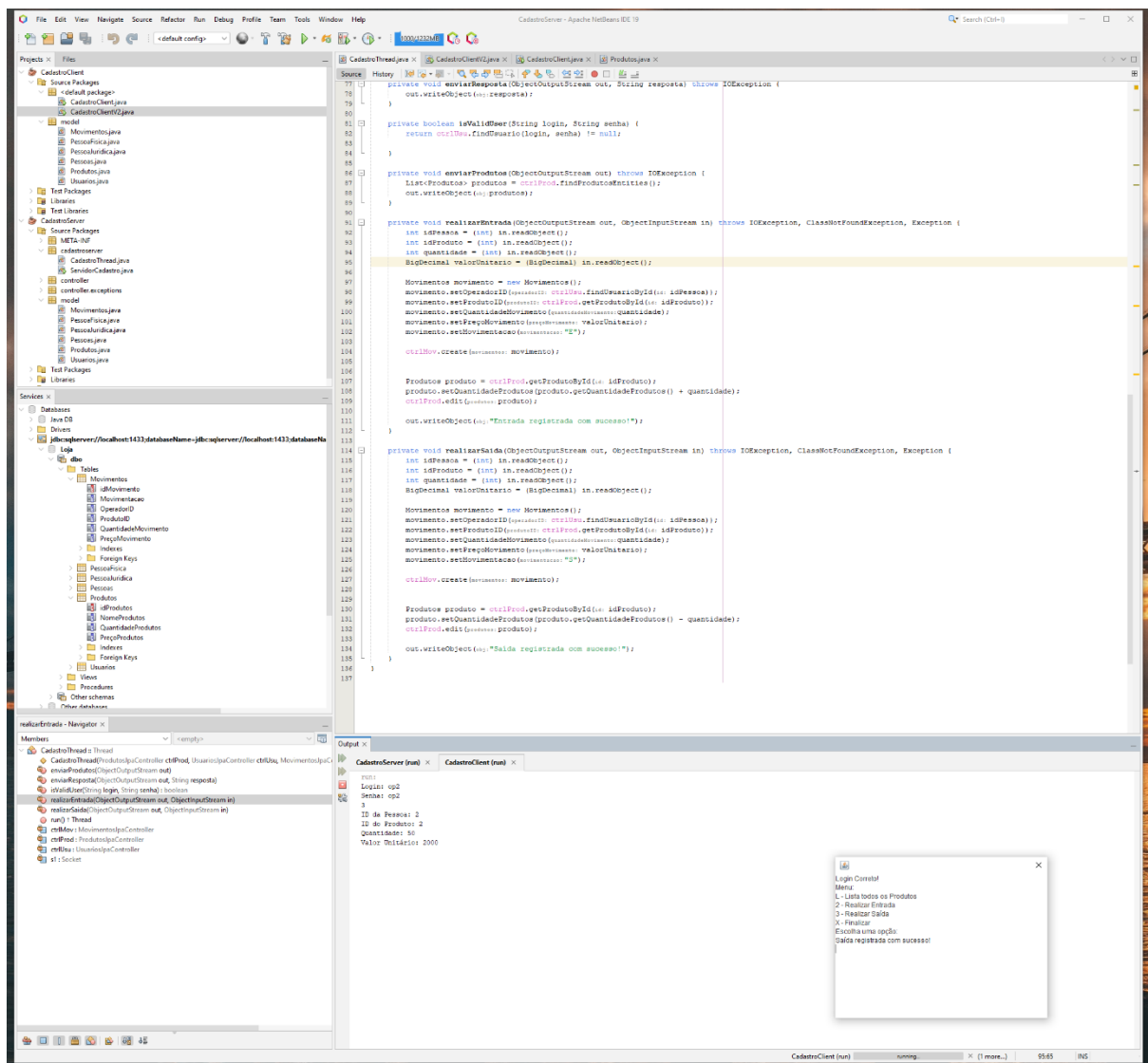


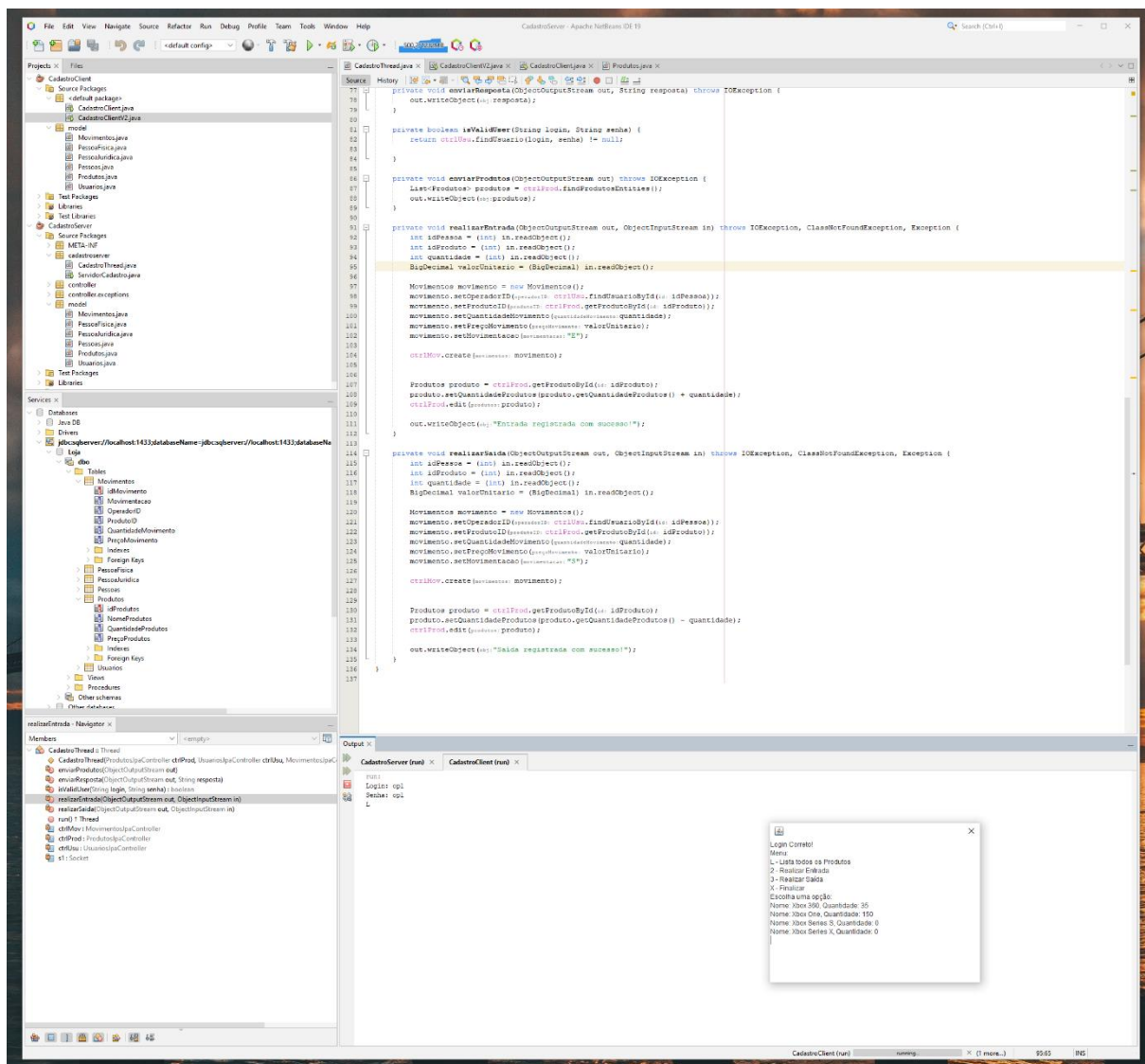
Resultados:











Análise e Conclusão:

Threads para Tratamento Assíncrono:

As threads são uma técnica fundamental para permitir o tratamento assíncrono das respostas enviadas pelo servidor em um sistema cliente-servidor. Quando um cliente se conecta a um servidor por meio de um soquete (socket) em Java, é criada uma conexão que permite a troca de informações entre as duas partes. No entanto, essa troca de informações pode ser síncrona ou assíncrona.

Para tratar respostas assíncronas, como mensagens do servidor que chegam em momentos imprevisíveis, geralmente é melhor usar threads. Isso permite que o programa cliente continue funcionando normalmente, executando suas operações, enquanto aguarda respostas do servidor em segundo plano. As threads permitem que o programa cliente execute múltiplas tarefas ao mesmo tempo, o que é crucial para aplicativos interativos que precisam responder a eventos em tempo real, como mensagens de chat ou atualizações de dados.

Método invokeLater da Classe SwingUtilities:

O método invokeLater da classe SwingUtilities é uma parte essencial do desenvolvimento de interfaces gráficas em Java. Ele é usado para garantir que a interface do usuário (UI) seja atualizada a partir da thread de eventos do Swing, que é a thread responsável por manipular componentes gráficos do Java Swing. A chamada direta a métodos que atualizam a interface a partir de uma thread não segura pode causar problemas de concorrência e travamentos na aplicação.

O invokeLater permite que você agende a execução de um pedaço de código na thread de eventos do Swing, garantindo que as operações de atualização da interface do usuário sejam executadas de forma segura. Isso é particularmente útil quando você recebe dados ou atualizações do servidor em uma thread diferente e precisa refletir essas alterações na interface gráfica.

Envio e Recebimento de Objetos pelo Socket Java:

O Socket em Java é uma abstração de comunicação que permite que os programas enviem e recebam dados pela rede. Para enviar e receber objetos por meio de sockets, os objetos precisam ser serializados em bytes antes de serem transmitidos pela rede e deserializados de volta para objetos na extremidade receptora.

Os métodos `ObjectOutputStream` e `ObjectInputStream` são usados para serializar e deserializar objetos, respectivamente. Os objetos são enviados como fluxos de bytes e podem ser recriados a partir desses fluxos na extremidade receptora. Isso permite que os programas enviem objetos complexos, como classes personalizadas, de um ponto a outro na rede.

Comportamento Assíncrono vs. Síncrono em Clientes com Socket Java:

A escolha entre comportamento assíncrono e síncrono em clientes com Socket Java depende das necessidades específicas da aplicação. O comportamento síncrono é mais simples de implementar e é adequado quando as operações podem ser bloqueantes e o cliente deve aguardar uma resposta do servidor antes de continuar. No entanto, pode resultar em uma experiência do usuário menos responsiva.

Por outro lado, o comportamento assíncrono, geralmente usando threads, permite que o cliente continue funcionando sem ser bloqueado por operações de rede. Isso é benéfico para aplicativos que precisam ser altamente responsivos. No entanto, requer gerenciamento adicional, como a sincronização de threads e a manipulação correta de eventos para garantir a consistência dos dados.

Em resumo, a escolha entre comportamento assíncrono e síncrono em clientes com Socket Java depende das necessidades da aplicação, priorizando a responsividade, a simplicidade de implementação e a garantia de que as operações de rede não bloqueiem excessivamente o aplicativo. Cada abordagem tem suas vantagens e desvantagens e deve ser escolhida com base nos requisitos específicos do projeto.