



# Estácio

Polo Tijuca

Aluno	Luiz Felipe Sarmiento Bonet
Matrícula	2023.09.34276-6
Curso	Desenvolvimento Full Stack
Disciplina	RPG0018 – Por que não paralelizar?
Turma	9001
Período	2024 – 2º Semestre

## Missão Prática: Mundo 3 - Nível 3

Título da Prática 1: Criando o Servidor e Cliente de Teste

Título da Prática 2: Servidor Completo e Cliente Assíncrono

### Objetivos:

- 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;
- Criar um servidor Java baseado em sockets, com acesso ao banco de dados via JPA, além de fazer uso dos recursos nativos do Java para implementação de clientes síncronos e assíncronos.

## Resultados de Execução – Prática 1

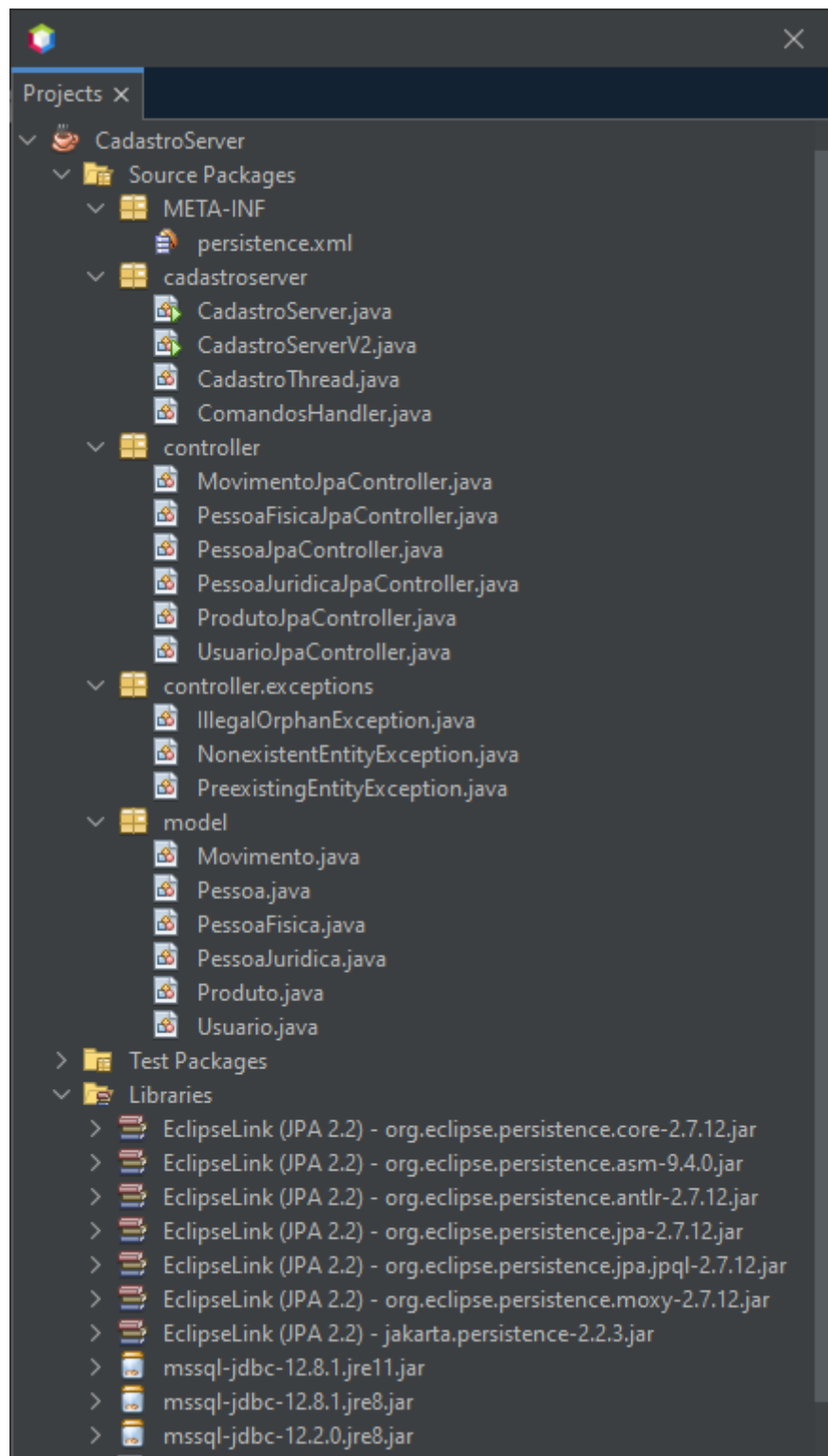


Figura 1. Aba de serviços do NetBeans mostrando os arquivos gerados durante procedimento de criação do CadastroServer.

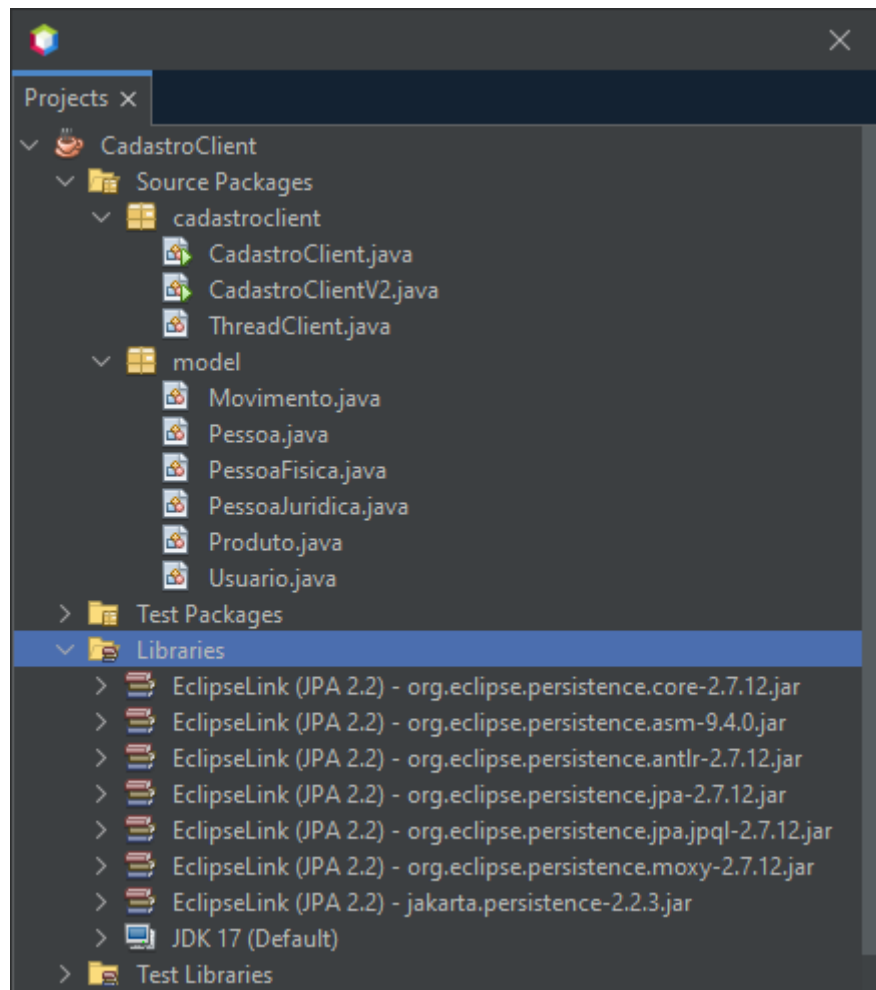


Figura 2. Aba de serviços do NetBeans mostrando os arquivos gerados durante procedimento de criação do CadastroClient.

```
run:
Aplicação conectada ao servidor Digite as informações de login:
Usuario: LojaGTJ
Senha: senha123
Login efetuado com sucesso. Exibindo produtos do banco de dados:
Placa Mae
Processador
Placa de Video
Fonte de Alimentacao
Memoria RAM
Monitor
Teclado
Mouse
BUILD SUCCESSFUL (total time: 1 minute 24 seconds)
```

Figura 3. Execução do CadastroClient após CadastroServer ter sido iniciado.

## Resultados de Execução – Prática 2

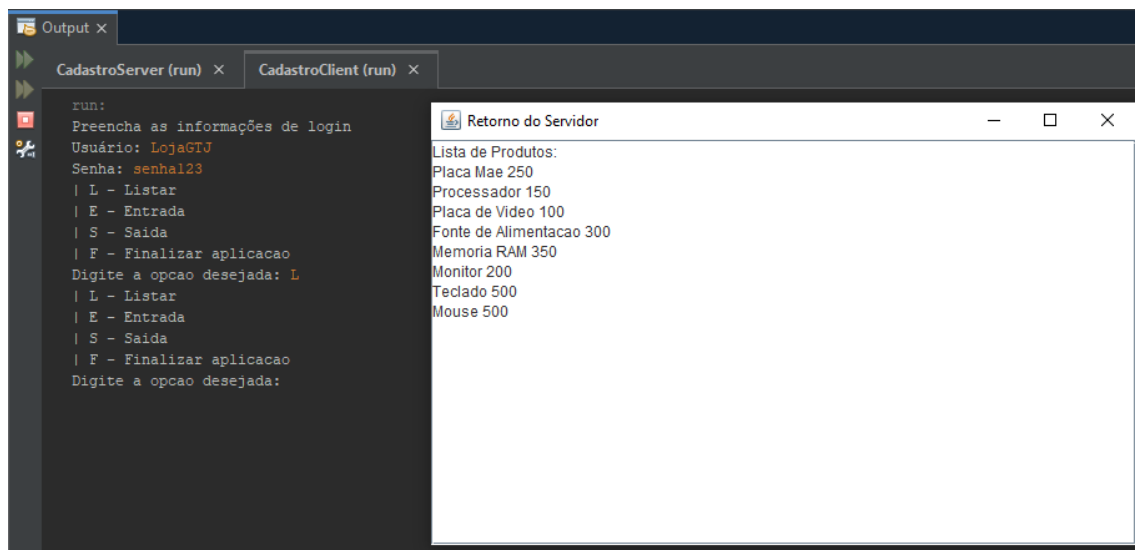


Figura 3. Execução de CadastroClientV2 após CadastroServerV2 ter sido iniciado, sendo feita uma operação de listagem de produtos.

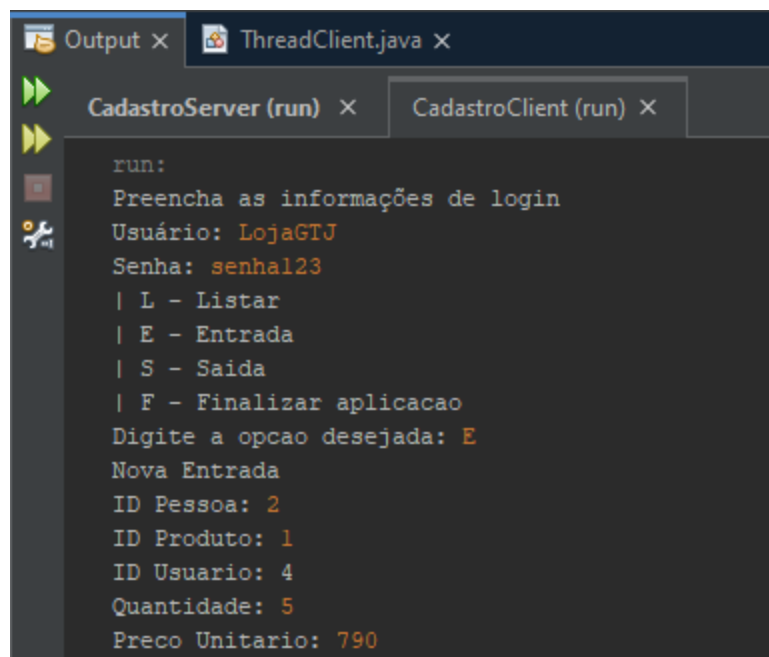
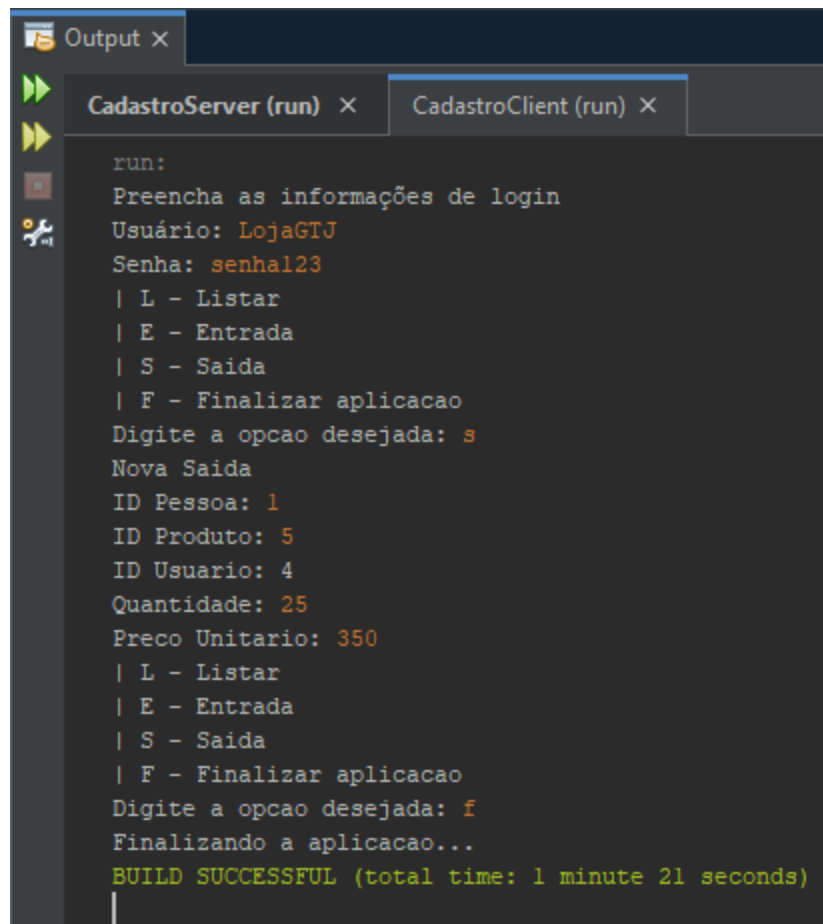


Figura 4. Console mostrando uma operação de entrada.



```
run:
Preencha as informações de login
Usuário: LojaGTJ
Senha: senha123
| L - Listar
| E - Entrada
| S - Saida
| F - Finalizar aplicacao
Digite a opcao desejada: s
Nova Saida
ID Pessoa: 1
ID Produto: 5
ID Usuario: 4
Quantidade: 25
Preco Unitario: 350
| L - Listar
| E - Entrada
| S - Saida
| F - Finalizar aplicacao
Digite a opcao desejada: f
Finalizando a aplicacao...
BUILD SUCCESSFUL (total time: 1 minute 21 seconds)
```

Figura 5. Console mostrando uma operação de saída e o encerramento da aplicação.

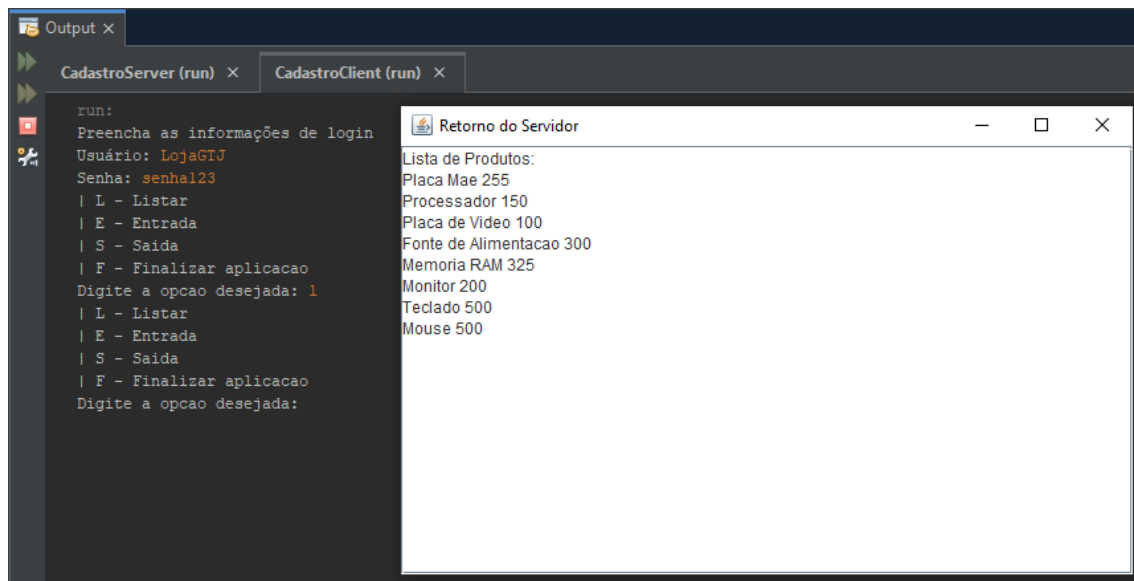


Figura 6. Retorno de um operação de listagem mostrando que as operações anteriores alteraram com sucesso as informações no banco de dados. Observe também que, pela simples adição de mais de uma variável (no caso a letra referente a cada operação, tanto em maiúscula quanto em minúscula) por *case* dentro do *switch*, o fato do usuário não utilizar a letra correspondente em maiúscula não causa problemas à aplicação.



## Análise e Conclusão – Prática 1

### **A) Como funcionam as classes *Socket* e *ServerSocket*?**

O *ServerSocket* é uma classe utilizada no lado do servidor que fica à espera de conexões para aceitar. Quando uma conexão é feita com sucesso, cria-se um objeto do tipo *Socket* que é usado para a comunicação entre servidor e cliente.

### **B) Qual a importância das portas para a conexão com servidores?**

As chamadas portas funcionam como endereços que estão associados a um processo ou serviço de um sistema. Ao especificar a porta numa conexão, o servidor, ou sistema operacional no lado cliente, sabe para onde direcionar o tráfego de rede recebido.

### **C) Para que servem as classes de entrada e saída *ObjectInputStream* e *ObjectOutputStream*, e por que os objetos transmitidos devem ser serializáveis?**

Ambas as classes são utilizadas para manipular a entrada e saída de objetos Java para fluxos de dados, bem como de fluxos de dados para objetos Java. Como os nomes indicam, *ObjectInputStream* é usado para converter os dados em objetos, enquanto *ObjectOutputStream* converte objetos em dados. É necessário que os objetos sejam serializáveis para que possam ser convertidos em sequências de bytes para poderem ser armazenados ou transmitidos.

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

Isso é possível pois, apesar da troca de dados entre servidor e cliente, este não acessa diretamente o banco de dados. As solicitações de operações são recebidas pelo servidor, que é quem faz todo o processamento dos dados e os envia para o cliente.

## Análise e Conclusão – Prática 2

### **A) Como as *threads* podem ser utilizadas para o tratamento assíncrono das respostas enviadas pelo servidor?**

Isso pode ser feito de maneira que cada solicitação seja processada por um thread diferente. Dessa forma, o servidor pode continuar aguardando por novas solicitações, enquanto solicitações já recebidas são processadas de forma paralela, sem que uma solicitação possa afetar toda a fila.

### **B) Para que serve o método *InvokeLater*, da classe *SwingUtilities*?**

Este método, mais utilizado na implementação de interfaces gráficas em Java, serve para otimizar o processo de atualização da interface gráfica, executando partes de código de forma assíncrona na *Event Dispatch Thread*. Com isso, ordenam-se as mudanças e evitam-se problemas na interface gráfica.

### **C) Como ao objetos são enviados e recebidos pelo *socket* Java?**

Através de *sockets* e *object I/O streams*. Primeiramente, um objeto é serializado e transformado em *stream*, através do *ObjectOutputStream*. Esse *stream* é transmitido para o outro lado através do *socket*, onde o *ObjectInputStream* lê o *stream* de dados, o desserializando e transformando de volta em objeto.

### **D) Compare a utilização de comportamento síncrono e assíncrono nos clientes com *socket* Java, ressaltando as características relacionadas ao bloqueio do processamento.**

Com o comportamento síncrono, as operações bloqueiam a fila de serviços até que a operação sendo processada no momento seja terminada, para que então outra operação na fila seja processada. Dessa forma, apesar de mais simples, pode ser um método menos eficiente para alguns tipos de tarefa, visto que se perde muito tempo em espera, enquanto cada serviço seja processado. Em contrapartida, com comportamento assíncrono, diferentes operações podem ser processadas de maneira paralela, sem bloquear a fila de serviços esperando que uma operação específica termine. Com isso, aumenta-se a rapidez e eficiência da aplicação.

## Anexo – Códigos do Projeto

### Arquivo – UsuarioJpaController.java

```
package controller;

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
 to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
 this template
 */

import controller.exceptions.IllegalOrphanException;
import controller.exceptions.NonexistentEntityException;
import java.io.Serializable;
import java.util.ArrayList;
import java.util.Collection;
import java.util.List;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityNotFoundException;
import javax.persistence.Query;
import model.Movimento;
import model.Usuario;

/**
 *
 *
 * @author fel-f

```

```

*/

public class UsuarioJpaController implements Serializable {

    public UsuarioJpaController(EntityManagerFactory emf) {
        this.emf = emf;
    }

    private EntityManagerFactory emf = null;

    public EntityManager getEntityManager() {
        return emf.createEntityManager();
    }

    public void create(Usuario usuario) {
        if (usuario.getMovimentoCollection() == null) {
            usuario.setMovimentoCollection(new ArrayList<Movimento>());
        }

        EntityManager em = null;
        try {
            em = getEntityManager();
            em.getTransaction().begin();

            Collection<Movimento> attachedMovimentoCollection = new
ArrayList<Movimento>();

            for (Movimento movimentoCollectionMovimentoToAttach :
usuario.getMovimentoCollection()) {
                movimentoCollectionMovimentoToAttach =
em.getReference(movimentoCollectionMovimentoToAttach.getClass(),
movimentoCollectionMovimentoToAttach.getIdMovimento());

                attachedMovimentoCollection.add(movimentoCollectionMovimentoToAttach);
            }
        }
    }
}

```

```

        usuario.setMovimentoCollection(attachedMovimentoCollection);

        em.persist(usuario);

        for (Movimento movimentoCollectionMovimento :
usuario.getMovimentoCollection()) {

            Usuario oldIdUsuarioOfMovimentoCollectionMovimento =
movimentoCollectionMovimento.getFKUsuarioid();

            movimentoCollectionMovimento.setFKUsuarioid(usuario);

            movimentoCollectionMovimento =
em.merge(movimentoCollectionMovimento);

            if (oldIdUsuarioOfMovimentoCollectionMovimento != null) {

oldIdUsuarioOfMovimentoCollectionMovimento.getMovimentoCollection().remo
ve(movimentoCollectionMovimento);

                oldIdUsuarioOfMovimentoCollectionMovimento =
em.merge(oldIdUsuarioOfMovimentoCollectionMovimento);

            }

        }

        em.getTransaction().commit();
    } finally {
        if (em != null) {
            em.close();
        }
    }
}

public void edit(Usuario usuario) throws IllegalOrphanException,
NonexistentEntityException, Exception {

    EntityManager em = null;

    try {
        em = getEntityManager();

        em.getTransaction().begin();

```

```

        Usuario persistentUsuario = em.find(Usuario.class,
usuario.getIdUsuario());

        Collection<Movimento> movimentoCollectionOld =
persistentUsuario.getMovimentoCollection();

        Collection<Movimento> movimentoCollectionNew =
usuario.getMovimentoCollection();

        List<String> illegalOrphanMessages = null;

        for (Movimento movimentoCollectionOldMovimento :
movimentoCollectionOld) {

            if
(!movimentoCollectionNew.contains(movimentoCollectionOldMovimento)) {

                if (illegalOrphanMessages == null) {

                    illegalOrphanMessages = new ArrayList<String>();

                }

                illegalOrphanMessages.add("You must retain Movimento " +
movimentoCollectionOldMovimento + " since its idUsuario field is not nullable.");

            }

        }

        if (illegalOrphanMessages != null) {

            throw new IllegalOrphanException(illegalOrphanMessages);

        }

        Collection<Movimento> attachedMovimentoCollectionNew = new
ArrayList<Movimento>();

        for (Movimento movimentoCollectionNewMovimentoToAttach :
movimentoCollectionNew) {

            movimentoCollectionNewMovimentoToAttach =
em.getReference(movimentoCollectionNewMovimentoToAttach.getClass(),
movimentoCollectionNewMovimentoToAttach.getIdMovimento());

            attachedMovimentoCollectionNew.add(movimentoCollectionNewMovimentoToA
ttach);

        }

        movimentoCollectionNew = attachedMovimentoCollectionNew;

```

```

        usuario.setMovimentoCollection(movimentoCollectionNew);

        usuario = em.merge(usuario);

        for (Movimento movimentoCollectionNewMovimento :
movimentoCollectionNew) {

            if
(!movimentoCollectionOld.contains(movimentoCollectionNewMovimento)) {

                Usuario oldIdUsuarioOfMovimentoCollectionNewMovimento =
movimentoCollectionNewMovimento.getFKUsuarioid();

                movimentoCollectionNewMovimento.setFKUsuarioid(usuario);

                movimentoCollectionNewMovimento =
em.merge(movimentoCollectionNewMovimento);

                if (oldIdUsuarioOfMovimentoCollectionNewMovimento != null &&
!oldIdUsuarioOfMovimentoCollectionNewMovimento.equals(usuario)) {

oldIdUsuarioOfMovimentoCollectionNewMovimento.getMovimentoCollection().r
emove(movimentoCollectionNewMovimento);

                oldIdUsuarioOfMovimentoCollectionNewMovimento =
em.merge(oldIdUsuarioOfMovimentoCollectionNewMovimento);

                }

            }

        }

        em.getTransaction().commit();
    } catch (Exception ex) {

        String msg = ex.getLocalizedMessage();

        if (msg == null || msg.length() == 0) {

            Integer id = usuario.getIdUsuario();

            if (findUsuario(id) == null) {

                throw new NonexistentEntityException("The usuario with id " + id +
" no longer exists.");

            }

        }

        throw ex;
    }

```

```

    } finally {
        if (em != null) {
            em.close();
        }
    }
}

public void destroy(Integer id) throws IllegalOrphanException,
NonexistentEntityException {
    EntityManager em = null;
    try {
        em = getEntityManager();
        em.getTransaction().begin();
        Usuario usuario;
        try {
            usuario = em.getReference(Usuario.class, id);
            usuario.getIdUsuario();
        } catch (EntityNotFoundException enfe) {
            throw new NonexistentEntityException("The usuario with id " + id + "
no longer exists.", enfe);
        }
        List<String> illegalOrphanMessages = null;
        Collection<Movimento> movimientoCollectionOrphanCheck =
usuario.getMovimentoCollection();
        for (Movimento movimientoCollectionOrphanCheckMovimento :
movimientoCollectionOrphanCheck) {
            if (illegalOrphanMessages == null) {
                illegalOrphanMessages = new ArrayList<String>();
            }
            illegalOrphanMessages.add("This Usuario (" + usuario + ") cannot be
destroyed since the Movimento " +

```



```
movimentoCollectionOrphanCheckMovimento + " in its movimientoCollection field has a non-nullable idUsuario field.");
```

```
    }  
    if (illegalOrphanMessages != null) {  
        throw new IllegalOrphanException(illegalOrphanMessages);  
    }  
    em.remove(usuario);  
    em.getTransaction().commit();  
} finally {  
    if (em != null) {  
        em.close();  
    }  
}  
}
```

```
public List<Usuario> findUsuarioEntities() {  
    return findUsuarioEntities(true, -1, -1);  
}
```

```
public List<Usuario> findUsuarioEntities(int maxResults, int firstResult) {  
    return findUsuarioEntities(false, maxResults, firstResult);  
}
```

```
private List<Usuario> findUsuarioEntities(boolean all, int maxResults, int firstResult) {
```

```
    EntityManager em = getEntityManager();  
    try {  
        CriteriaQuery cq = em.getCriteriaBuilder().createQuery();  
        cq.select(cq.from(Usuario.class));  
        Query q = em.createQuery(cq);
```

```

        if (!all) {
            q.setMaxResults(maxResults);
            q.setFirstResult(firstResult);
        }
        return q.getResultList();
    } finally {
        em.close();
    }
}

```

```

public Usuario findUsuario(Integer id) {
    EntityManager em = getEntityManager();
    try {
        return em.find(Usuario.class, id);
    } finally {
        em.close();
    }
}

```

```

public int getUsuarioCount() {
    EntityManager em = getEntityManager();
    try {
        CriteriaQuery cq = em.getCriteriaBuilder().createQuery();
        Root<Usuario> rt = cq.from(Usuario.class);
        cq.select(em.getCriteriaBuilder().count(rt));
        Query q = em.createQuery(cq);
        return ((Long) q.getSingleResult()).intValue();
    } finally {
        em.close();
    }
}

```

```

    }
}

public Usuario validarUsuario(String login, String senha) {
    EntityManager em = getEntityManager();
    try {
        Query query = em.createQuery("SELECT u FROM Usuario u WHERE
u.login = :login AND u.senha = :senha");
        query.setParameter("login", login);
        query.setParameter("senha", senha);

        List<Usuario> resultados = query.getResultList();
        return resultados.isEmpty() ? null : resultados.get(0);
    } finally {
        em.close();
    }
}
}
}

```

#### Arquivo – CadastroServer.java

```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit
this template
 */

package cadastroserver;

```

```

import controller.MovimentoJpaController;
import controller.PessoaJpaController;
import controller.ProdutoJpaController;
import controller.UsuarioJpaController;
import model.Produto;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.List;
import java.util.logging.Logger;
import java.util.logging.Level;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

/**
 *
 * @author fel-f
 */

public class CadastroServer {

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

```

```

private final int PORT = 4321;

public CadastroServer() {

}

private void run() {
    try (ServerSocket serverSocket = new ServerSocket(PORT)) {
        System.out.println("Servidor conectado em:" + PORT);
        // Inicializa controladores
        EntityManagerFactory emf =
Persistence.createEntityManagerFactory("CadastroServerPU");
        ProdutoJpaController produtoController = new
ProdutoJpaController(emf);
        MovimentoJpaController movimentoController = new
MovimentoJpaController(emf);
        PessoaJpaController pessoaController = new
PessoaJpaController(emf);
        UsuarioJpaController usuarioController = new
UsuarioJpaController(emf);
        while (true) {
            System.out.println("Aguardando conexao de cliente...");
            Socket socket = serverSocket.accept();
            System.out.println("Usuario conectado.");
            ClientHandler clientHandler = new ClientHandler(socket,
produtoController,
            movimentoController, pessoaController, usuarioController);
            Thread thread = new Thread(clientHandler);
            thread.start();
        }
    }
}

```

```

        } catch (IOException e) {

            Logger.getLogger(CadastroServer.class.getName()).log(Level.SEVERE,
null, e);

        }

    }

    public static void main(String[] args) {

        new CadastroServer().run();

    }

    private class ClientHandler implements Runnable {

        private final Socket socket;

        private final ProdutoJpaController produtoController;

        private final UsuarioJpaController usuarioController;

        public ClientHandler(Socket socket, ProdutoJpaController
produtoController,

            MovimentoJpaController movimentoController, PessoaJpaController
pessoaController,

            UsuarioJpaController usuarioController) {

            this.socket = socket;

            this.produtoController = produtoController;

            this.usuarioController = usuarioController;

        }

        @Override

        public void run() {

            try (

                BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));

```

```

        PrintWriter out = new PrintWriter(socket.getOutputStream(), true)
    ) {
        // Autenticacao

        String username = in.readLine();

        String password = in.readLine();

        if (validateCredentials(username, password)) {

            out.println("Login efetuado com sucesso. Exibindo produtos do
banco de dados:");

            sendProductList(out);
        } else {
            try (socket) {
                out.println("Ocorreu um erro durante o login.");
            }
        }
    } catch (IOException e) {

Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE, null, e);
    } catch (Exception e) {

Logger.getLogger(ClientHandler.class.getName()).log(Level.SEVERE, null, e);
    }
}

private boolean validateCredentials(String username, String password) {
    return usuarioController.validarUsuario(username, password) != null;
}

private void sendProductList(PrintWriter out) {
    List<Produto> productList = produtoController.findProdutoEntities();
    for (Produto product : productList) {

```

```

        out.println(product.getNome());
    }
    out.println();
}

}

}

```

#### Arquivo – CadastroClient.java

```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit
this template
 */

package cadastroclient;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;
import java.util.logging.Logger;
import java.util.logging.Level;

/**
 *
 * @author fel-f
 */

```



```

*/

public class CadastroClient {

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

    private final String ADDRESS = "localhost";
    private final int PORT = 4321;

    public CadastroClient() {

    }

    private void run() {
        try (Socket socket = new Socket(ADDRESS, PORT);
            BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
            BufferedReader consoleIn = new BufferedReader(new
InputStreamReader(System.in))) {
            System.out.println("Aplicação conectada ao servidor Digite as
informações de login:");
            System.out.print("Usuario: ");
            String username = consoleIn.readLine();
            System.out.print("Senha: ");
            String password = consoleIn.readLine();
            out.println(username);
            out.println(password);
        }
    }
}

```

```

        String response = in.readLine();

        System.out.println(response);

        if (response.equals("Login efetuado com sucesso. Exibindo produtos do
banco de dados:")) {
            receiveAndDisplayProductList(in);
        } else {
            System.out.println("Ocorreu um erro durante o login.");
        }
    } catch (IOException e) {
        Logger.getLogger(CadastroClient.class.getName()).log(Level.SEVERE,
null, e);
    }
}

private void receiveAndDisplayProductList(BufferedReader in) throws
IOException {
    String line;
    while ((line = in.readLine()) != null && !line.isEmpty()) {
        System.out.println(line);
    }
}

public static void main(String[] args) {
    new CadastroClient().run();
}
}

```

Arquivo – CadastroThread.java

/\*

\* Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license

\* Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template

\*/

```
package cadastrserver;
```

```
import controller.UsuarioJpaController;
```

```
import model.Usuario;
```

```
import java.io.IOException;
```

```
import java.io.ObjectInputStream;
```

```
import java.io.ObjectOutputStream;
```

```
import java.net.Socket;
```

```
import java.text.SimpleDateFormat;
```

```
import java.util.Date;
```

```
import java.util.logging.Level;
```

```
import java.util.logging.Logger;
```

```
/**
```

```
 *
```

```
 * @author fel-f
```

```
 */
```

```
public class CadastroThread extends Thread {
```

```
    private final UsuarioJpaController ctrlUsu;
```

```
    private final Socket s1;
```

```

public CadastroThread(UsuarioJpaController ctrlUsu, Socket s1) {

    this.ctrlUsu = ctrlUsu;

    this.s1 = s1;

}

@Override

public void run() {

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

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

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

        Date dataAtual = new Date();

        SimpleDateFormat formato = new SimpleDateFormat("dd/MM/yyyy
HH:mm:ss");

        String dataFormatada = formato.format(dataAtual);

        System.out.println("Nova comunicacao em -> " + dataFormatada);

        boolean usuarioValido = (validar(login, senha) != null);

        if (usuarioValido) {

            out.writeObject(usuarioValido);

            out.writeObject(validar(login, senha).getIdUsuario());

            System.out.println("Login bem sucedido!");

            ComandosHandler comandos = new ComandosHandler(out, in);

            comandos.executarComandos();

```

```

        } else {
            out.writeObject(usuarioValido);

            out.writeObject(null);

        }
        out.flush();

    } catch (IOException | ClassNotFoundException ex) {

        Logger.getLogger(CadastroThread.class.getName()).log(Level.SEVERE, null,
ex);
    }
}

private Usuario validar(String login, String senha) {
    return ctrlUsu.validarUsuario(login, senha);
}
}

```

#### Arquivo – CadastroServerV2.java

```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit
this template
 */

```

```

package cadastroserver;

import controller.UsuarioJpaController;
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

/**
 *
 * @author fel-f
 */
public class CadastroServerV2 {

    private final int PORT = 4321;

    public CadastroServerV2() {

    }

    private void run() {
        EntityManagerFactory emf =
Persistence.createEntityManagerFactory("CadastroServerPU");
        UsuarioJpaController ctrlUsu = new UsuarioJpaController(emf);
        try (ServerSocket serverSocket = new ServerSocket(PORT)) {

```

```

        System.out.println("Servidor conectado em: " + PORT);
        while (true) {
            Socket socket = serverSocket.accept();
            CadastroThread teste = new CadastroThread(ctrlUsu, socket);
            teste.start();
            System.out.println("A thread foi iniciada com sucesso!");
        }
    } catch (IOException ex) {

        Logger.getLogger(CadastroServerV2.class.getName()).log(Level.SEVERE, null,
ex);
    }
}

/**
 * @param args the command line arguments
 */
public static void main(String[] args) {
    new CadastroServerV2().run();
}
}

```

#### Arquivo – CadastroClientV2.java

```

/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit
this 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.net.Socket;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author fel-f
 */

public class CadastroClientV2 {

    private final String ADDRESS = "localhost";
    private final int PORT = 4321;

    public CadastroClientV2() {

    }

    private void run() {
        try {
            Socket socket = new Socket(ADDRESS, PORT);
```



```

        ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());

        ObjectInputStream in = new
ObjectInputStream(socket.getInputStream());

        BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));

        System.out.println("Preencha as informações de login");

        System.out.print("Usuário: ");

        String login = reader.readLine();

        System.out.print("Senha: ");

        String senha = reader.readLine();

        out.writeObject(login);

        out.writeObject(senha);

        out.flush();

        ThreadClient threadClient = new ThreadClient(in,out);

        threadClient.start();

    } catch (IOException ex) {

Logger.getLogger(CadastroClientV2.class.getName()).log(Level.SEVERE, null,
ex);

    }

}

/**
 * @param args the command line arguments
 */
public static void main(String[] args) {

    new CadastroClientV2().run();

}

}

```

## Arquivo – ThreadClient.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt
 to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit
 this template
 */

package cadastroclient;

import java.awt.HeadlessException;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.util.ArrayList;
import javax.swing.JFrame;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;
import javax.swing.SwingUtilities;

/**
 *
 * @author fel-f
 */

public class ThreadClient extends Thread {
```

```

private ObjectOutputStream out = null;
private ObjectInputStream in = null;

private JTextArea textArea;
private JFrame frame;
private ArrayList<String> output;

public ThreadClient() {

}

public ThreadClient(ObjectInputStream in, ObjectOutputStream out) {
    this.in = in;
    this.out = out;
}

@Override
public void run() {
    output = new ArrayList<>();
    try {
        Boolean validate = (Boolean) in.readObject();
        Integer idUsuario = (Integer) in.readObject();

        if (validate) {
            BufferedReader reader = new BufferedReader(new
InputStreamReader(System.in));

            String comando;
            String idPessoa;

```

```
String idProduto;

String quantidade;

String precoUnitario;

do {

    System.out.println("| L - Listar");
    System.out.println("| E - Entrada");
    System.out.println("| S - Saida");
    System.out.println("| F - Finalizar aplicacao");
    System.out.print("Digite a opcao desejada: ");
    comando = reader.readLine();

    switch (comando) {
        case "E", "e" -> {
            out.writeObject("E");

            System.out.println("Nova Entrada");

            System.out.print("ID Pessoa: ");
            idPessoa = reader.readLine();
            out.writeObject(idPessoa);

            System.out.print("ID Produto: ");
            idProduto = reader.readLine();
            out.writeObject(idProduto);

            System.out.print("ID Usuario: " + idUsuario);
            out.writeObject(idUsuario);
            System.out.println("");
        }
    }
}
```

```
System.out.print("Quantidade: ");
quantidade = reader.readLine();
out.writeObject(quantidade);

System.out.print("Preco Unitario: ");
precoUnitario = reader.readLine();
out.writeObject(precoUnitario);

output.add("Entrada realizada com sucesso.\n");
}

case "S", "s" -> {
    out.writeObject("S");

    System.out.println("Nova Saida");

    System.out.print("ID Pessoa: ");
    idPessoa = reader.readLine();
    out.writeObject(idPessoa);

    System.out.print("ID Produto: ");
    idProduto = reader.readLine();
    out.writeObject(idProduto);

    System.out.print("ID Usuario: " + idUsuario);
    out.writeObject(idUsuario);
    System.out.println("");
}
```

```

        System.out.print("Quantidade: ");
        quantidade = reader.readLine();
        out.writeObject(quantidade);

        System.out.print("Preco Unitario: ");
        precoUnitario = reader.readLine();
        out.writeObject(precoUnitario);

        output.add("Saida realizada com sucesso.\n");
    }

    case "L", "I" -> {
        out.writeObject("L");
        try {
            ArrayList<String> produtoList = (ArrayList<String>)
in.readObject();
            ArrayList<Integer> produtoQuantidade =
(ArrayList<Integer>) in.readObject();
            if (frame == null || !frame.isVisible()) {
                frame = new JFrame("Retorno do Servidor");
                frame.setSize(400, 600);
                textArea = new JTextArea(20, 50);
                textArea.setEditable(false);
                frame.add(new JScrollPane(textArea));
                frame.pack();

frame.setDefaultCloseOperation(JFrame.HIDE_ON_CLOSE);

                frame.setVisible(true);
                frame.setVisible(true);
                SwingUtilities.invokeLater(() -> {

```

```

        output.add("Lista de Produtos:\n");
        for (int i = 0; i < produtoList.size(); i++) {
            output.add(produtoList.get(i) + " " +
produtoQuantidade.get(i) + "\n");
        }
        for (String line : output) {
            textArea.append(line);
        }

textArea.setCaretPosition(textArea.getDocument().getLength());

    });
    } else {
        frame.setVisible(false);
    }

    } catch (ClassNotFoundException | IOException e) {
        e.printStackTrace();
    }
}

case "F", "f" -> {
    out.writeObject("F");
    System.out.println("Finalizando a aplicacao...");
}

default -> System.out.println("Por favor, digite uma opcao
valida.");
}

} while (!"f".equalsIgnoreCase(comando));

```

```
    } else {  
        System.out.println("Usuario ou senha incorretos!");  
    }  
  
} catch (HeadlessException | IOException | ClassNotFoundException e) {  
    if (!(e instanceof java.io.EOFException)) {  
        System.out.println("Finalizando a thread...");  
    }  
}  
}  
}
```