UNIVERSIDADE ESTÁCIO DE SÁ

Campus Estácio - Castelo - Belo Horizonte - MG

Curso: Desenvolvimento Full Stack

Disciplina: Nível 5: Por que não paralelizar? (RPG0018)

Turma: 9002

Semestre Letivo: 2025.1

Missão Prática | Nível 5 | Mundo 3

1º Procedimento | Criando o Servidor e Cliente de Teste

Aluno: Ivan de Ávila Carvalho Fleury Mortimer

Repositório dos Códigos no GitHub:

Data: 10 de junho de 2025

Índice

Table of Contents

Missão Prática | Nível 5 | Mundo 3

1. Introdução

O 1º Procedimento da Missão Prática | Nível 5 | Mundo 3 teve como objetivo principal a criação de um servidor e um cliente de teste com comunicação baseada em Sockets Java, utilizando Threads para o atendimento simultâneo de clientes e com acesso a banco de dados via JPA.

O servidor, denominado CadastroServer, foi implementado para:

- aceitar conexões de clientes via ServerSocket;
- autenticar usuários com base na tabela Usuario;
- responder a comandos simples (como a solicitação da lista de produtos);
- atender cada cliente em uma Thread separada.

O cliente, denominado CadastroClient, foi implementado para:

- conectar-se ao servidor;
- realizar o processo de autenticação;
- enviar comandos e receber respostas do servidor.

A prática exigiu a aplicação de conceitos fundamentais de conexões de rede, persistência com JPA e programação concorrente em Java.

2. Objetivos da prática

1. Criar servidores Java com base em Sockets.
2. Criar clientes síncronos para servidores com base em Sockets.
3. Criar clientes assíncronos para servidores com base em Sockets.
4. Utilizar Threads para implementação de processos paralelos.

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. Resultados esperados para o 1° Procedimento

1. É importante que o código seja organizado.
2. Outro ponto importante é explorar as funcionalidades oferecidas pelo NetBeans para melhoria da produtividade.
3. Nesse exercício, é esperado que o estudante demonstre as habilidades básicas no uso prático de Threads em ambientes cliente e servidor.

4. Execução e Resultados do Sistema

Execução do Servidor

O projeto CadastroServer foi executado com sucesso. A seguinte mensagem foi apresentada no console, indicando que o servidor está ativo e escutando na porta 4321:

```
INFO: CadastroServer iniciado na porta 4321.
```

- **Execução do Cliente**

  O projeto CadastroClient foi executado em seguida. A saída no console foi exatamente conforme especificação da missão:

```
run:
Usuario conectado com sucesso
Banana
Laranja
Manga
BUILD SUCCESSFUL (total time: 2 seconds)
```

- **Log do Servidor durante a execução**

  Durante a execução do cliente, o servidor apresentou no console as seguintes informações:

```
Cliente conectado: 127.0.0.1
Usuário autenticado: op1
Lista de produtos enviada para: op1
Conexão encerrada para usuário: op1
```

5. Observações Finais

Durante a implementação e execução do 1º Procedimento, foi possível validar com sucesso os seguintes aspectos:

- A correta integração entre o servidor multithreaded e o cliente de teste;
- A utilização de Threads no servidor, permitindo múltiplos clientes paralelos;
- A persistência de dados no banco de dados loja, com acesso via JPA;
- A correta serialização e desserialização dos objetos JPA (Produto) entre servidor e cliente.

A comunicação foi estabelecida com sucesso, e o comportamento funcional do sistema esteve em total conformidade com os requisitos descritos no roteiro da missão prática.

6. Análise e Conclusão

a. Como funcionam as classes Socket e ServerSocket?

As classes Socket e ServerSocket fazem parte da API de rede do Java e permitem a comunicação entre processos através do protocolo TCP/IP.

A classe ServerSocket é utilizada pelo lado servidor para "escutar" requisições de conexão

em uma porta específica. A classe Socket é utilizada pelo lado cliente para estabelecer uma conexão com um servidor que esteja escutando em uma porta específica. Uma vez estabelecida a conexão, ambos (cliente e servidor) podem trocar dados utilizando fluxos de entrada e saída.

b. Qual a importância das portas para a conexão com servidores?

As portas permitem que múltiplas aplicações em um mesmo computador possam utilizar a rede simultaneamente. O ServerSocket se associa a uma porta específica, e é através dessa porta que clientes sabem onde encontrar o servidor. A correta escolha e gestão das portas é essencial para evitar conflitos e garantir comunicação previsível e segura.

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

As classes ObjectInputStream e ObjectOutputStream permitem a transmissão de objetos Java inteiros pela rede. ObjectOutputStream converte (serializa) um objeto em um fluxo de bytes; ObjectInputStream reconstrói (desserializa) o objeto original. Para que isso funcione, os objetos devem implementar a interface Serializable.

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

O isolamento foi garantido porque o cliente não acessa diretamente o banco de dados. Ele apenas recebe objetos instanciados e preparados pelo servidor. O servidor é o único componente que possui a EntityManagerFactory e que executa consultas e comandos JPA. No cliente, as classes de entidade são utilizadas apenas como estruturas de dados.

7. Códigos fonte completos e seu repositório no GitHub

Todos os códigos fonte a seguir podem ser encontrados no seguinte repositório do GitHub:

https://github.com/ivanmortimer/CadastroSocketServerClient_MP_N5_M3

7.1. Códigos fonte do projeto de aplicativo Java (with Ant) “CadastroServer”

7.1.1. Arquivo pacote Java “META-INF” do projeto “CadastroServer”

7.1.1.1. Arquivo “persistence.xml”

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.2"
xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_2.xsd">
  <persistence-unit name="CadastroServerPU" transaction-
type="RESOURCE_LOCAL">

<provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>
    <class>model.Pessoa</class>
    <class>model.Movimento</class>
    <class>model.Usuario</class>
    <class>model.Produto</class>
    <class>model.PessoaJuridica</class>
    <class>model.PessoaFisica</class>
    <properties>
      <property name="javax.persistence.jdbc.url"
value="jdbc:sqlserver://localhost:1433;databaseName=loja;encrypt=true;t
rustServerCertificate=true"/>
      <property name="javax.persistence.jdbc.user" value="loja"/>
      <property name="javax.persistence.jdbc.driver"
value="com.microsoft.sqlserver.jdbc.SQLServerDriver"/>
      <property name="javax.persistence.jdbc.password" value="loja"/>
      <!-- Important: prevent JPA from trying to recreate the schema --
>
      <property name="javax.persistence.schema-
generation.database.action" value="none"/>
      <!-- Useful EclipseLink properties -->
      <property name="eclipselink.logging.level" value="FINE"/>
      <property name="eclipselink.target-database" value="SQLServer"/>
    </properties>
  </persistence-unit>
</persistence>
```

7.1.2. Arquivos do pacote Java “cadastroserver” do projeto “CadastroServer”

7.1.2.1. Arquivo “CadastroServerMain.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.ProdutoJpaController;
import controller.UsuarioJpaController;
import java.io.IOException;

import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Ivan
 */
public class CadastroServerMain {

    private static final Logger LOGGER =
Logger.getLogger(CadastroServerMain.class.getName());

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        try {
            // (1) Instanciar EntityManagerFactory
            EntityManagerFactory emf =
```

```
Persistence.createEntityManagerFactory("CadastroServerPU");

            // (2) Instanciar os controllers
            ProdutoJpaController ctrlProduto = new
ProdutoJpaController(emf);
            UsuarioJpaController ctrlUsuario = new
UsuarioJpaController(emf);

            // (3) Criar o ServerSocket na porta 4321
            ServerSocket serverSocket = new ServerSocket(4321);
            LOGGER.info("CadastroServer iniciado na porta 4321.");

            // (4) Loop infinito de atendimento
            while (true) {
                // Esperar conexão de cliente
                Socket clientSocket = serverSocket.accept();
                LOGGER.log(Level.INFO, "Cliente conectado: {0}",
clientSocket.getInetAddress().getHostAddress());

                // Instanciar thread para atendimento do cliente
                CadastroThread thread = new CadastroThread(ctrlProduto,
ctrlUsuario, clientSocket);
                thread.start();
            }

        } catch (IOException e) {
            LOGGER.log(Level.SEVERE, "Erro no servidor", e);
        }
    }
}
```

7.1.2.2. 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 cadastroserver;
```

```
import controller.ProdutoJpaController;
import controller.UsuarioJpaController;
import model.Produto;
import model.Usuario;

import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.io.Serializable;
import java.net.Socket;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Ivan
 */
public class CadastroThread extends Thread implements Serializable {

    private static final long serialVersionUID = 1L;

    private static final Logger LOGGER =
Logger.getLogger(CadastroThread.class.getName());

    private final ProdutoJpaController ctrl;
    private final UsuarioJpaController ctrlUsu;
    private final Socket s1;

    public CadastroThread(ProdutoJpaController ctrl,
UsuarioJpaController ctrlUsu, Socket s1) {
        this.ctrl = ctrl;
        this.ctrlUsu = ctrlUsu;
        this.s1 = s1;
    }

    @Override
    public void run() {
        try (
            ObjectOutputStream out = new
```

```
ObjectOutputStream(s1.getOutputStream());
            ObjectInputStream in = new
ObjectInputStream(s1.getInputStream());
        ) {
            // (1) Receber login e senha
            String login = (String) in.readObject();
            String senha = (String) in.readObject();

            // (2) Validar usuário
            Usuario usuario = ctrlUsu.findUsuario(login, senha);

            if (usuario == null) {
                // Usuário inválido → encerrar conexão
                LOGGER.log(Level.INFO, "Login inválido: {0}", login);
                s1.close();
                return;
            }

            LOGGER.log(Level.INFO, "Usuário autenticado: {0}", login);

            // (3) Loop de comandos
            boolean running = true;
            while (running) {
                String comando = (String) in.readObject();

                if (comando == null) {
                    running = false;
                    continue;
                }

                switch (comando) {
                    case "L" -> {
                        // Enviar lista de produtos
                        List<Produto> produtos =
ctrl.findProdutoEntities();
                        out.writeObject(produtos);
                        out.flush();
                        LOGGER.log(Level.INFO, "Lista de produtos
enviada para: {0}", login);
                    }
```

```
                    default -> {
                        // Comando não reconhecido → encerrar conexão
                        running = false;
                    }
                }
            }

            // (4) Fechar conexão
            s1.close();
            LOGGER.log(Level.INFO, "Conexão encerrada para usuário:
{0}", login);

        } catch (Exception e) {
            LOGGER.log(Level.SEVERE, "Erro na thread de atendimento",
e);
        }
    }
}
```

7.1.3. Arquivos do pacote Java "controller" do projeto "CadastroServer"

7.1.3.1. Arquivo "MovimentoJpaController.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 controller;

import controller.exceptions.NonexistentEntityException;
import model.Movimento;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Query;
```

```
import javax.persistence.EntityNotFoundException;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;
import java.io.Serializable;
import java.util.List;

/**
 *
 * @author Ivan
 */
public class MovimentoJpaController implements Serializable {

    private static final long serialVersionUID = 1L;

    private EntityManagerFactory emf = null;

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

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

    public void create(Movimento movimento) {
        EntityManager em = getEntityManager();
        try {
            em.getTransaction().begin();
            em.persist(movimento);
            em.getTransaction().commit();
        } finally {
            em.close();
        }
    }

    public void edit(Movimento movimento) throws
NonexistentEntityException, Exception {
        EntityManager em = getEntityManager();
        try {
```

```
            em.getTransaction().begin();
            movimento = em.merge(movimento);
            em.getTransaction().commit();
        } catch (Exception ex) {
            Integer id = movimento.getIdMovimento();
            if (findMovimento(id) == null) {
                throw new NonexistentEntityException("The movimento
with id " + id + " no longer exists.");
            }
            throw ex;
        } finally {
            em.close();
        }
    }

    public void destroy(Integer id) throws NonexistentEntityException {
        EntityManager em = getEntityManager();
        try {
            em.getTransaction().begin();
            Movimento movimento;
            try {
                movimento = em.getReference(Movimento.class, id);
                movimento.getIdMovimento();
            } catch (EntityNotFoundException enfe) {
                throw new NonexistentEntityException("The movimento
with id " + id + " no longer exists.", enfe);
            }
            em.remove(movimento);
            em.getTransaction().commit();
        } finally {
            em.close();
        }
    }

    public List<Movimento> findMovimentoEntities() {
        return findMovimentoEntities(true, -1, -1);
    }

    public List<Movimento> findMovimentoEntities(int maxResults, int
firstResult) {
```

```
        return findMovimentoEntities(false, maxResults, firstResult);
    }

    private List<Movimento> findMovimentoEntities(boolean all, int
maxResults, int firstResult) {
        EntityManager em = getEntityManager();
        try {
            CriteriaQuery cq = em.getCriteriaBuilder().createQuery();
            cq.select(cq.from(Movimento.class));
            Query q = em.createQuery(cq);
            if (!all) {
                q.setMaxResults(maxResults);
                q.setFirstResult(firstResult);
            }
            return q.getResultList();
        } finally {
            em.close();
        }
    }

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

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

```
        }
    }
}
```

7.1.3.2. Arquivo "PessoaJpaController.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 controller;

import controller.exceptions.NonexistentEntityException;
import model.Pessoa;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Query;
import javax.persistence.EntityNotFoundException;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;
import java.io.Serializable;
import java.util.List;

/**
 *
 * @author Ivan
 */
public class PessoaJpaController implements Serializable {

    private static final long serialVersionUID = 1L;

    private EntityManagerFactory emf = null;

    public PessoaJpaController(EntityManagerFactory emf) {
        this.emf = emf;
    }
```

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

    public void create(Pessoa pessoa) {
        EntityManager em = getEntityManager();
        try {
            em.getTransaction().begin();
            em.persist(pessoa);
            em.getTransaction().commit();
        } finally {
            em.close();
        }
    }

    public void edit(Pessoa pessoa) throws NonexistentEntityException,
Exception {
        EntityManager em = getEntityManager();
        try {
            em.getTransaction().begin();
            pessoa = em.merge(pessoa);
            em.getTransaction().commit();
        } catch (Exception ex) {
            Integer id = pessoa.getIdPessoa();
            if (findPessoa(id) == null) {
                throw new NonexistentEntityException("The pessoa with
id " + id + " no longer exists.");
            }
            throw ex;
        } finally {
            em.close();
        }
    }

    public void destroy(Integer id) throws NonexistentEntityException {
        EntityManager em = getEntityManager();
        try {
            em.getTransaction().begin();
```

```
            Pessoa pessoa;
            try {
                pessoa = em.getReference(Pessoa.class, id);
                pessoa.getIdPessoa();
            } catch (EntityNotFoundException enfe) {
                throw new NonexistentEntityException("The pessoa with
id " + id + " no longer exists.", enfe);
            }
            em.remove(pessoa);
            em.getTransaction().commit();
        } finally {
            em.close();
        }
    }

    public List<Pessoa> findPessoaEntities() {
        return findPessoaEntities(true, -1, -1);
    }

    public List<Pessoa> findPessoaEntities(int maxResults, int
firstResult) {
        return findPessoaEntities(false, maxResults, firstResult);
    }

    private List<Pessoa> findPessoaEntities(boolean all, int
maxResults, int firstResult) {
        EntityManager em = getEntityManager();
        try {
            CriteriaQuery cq = em.getCriteriaBuilder().createQuery();
            cq.select(cq.from(Pessoa.class));
            Query q = em.createQuery(cq);
            if (!all) {
                q.setMaxResults(maxResults);
                q.setFirstResult(firstResult);
            }
            return q.getResultList();
        } finally {
            em.close();
        }
    }
```

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

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

7.1.3.3. Arquivo "ProdutoJpaController.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 controller;

import controller.exceptions.NonexistentEntityException;
import model.Produto;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
```

```
import javax.persistence.Query;
import javax.persistence.EntityNotFoundException;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;
import java.io.Serializable;
import java.util.List;

/**
 *
 * @author Ivan
 */
public class ProdutoJpaController implements Serializable {

    private static final long serialVersionUID = 1L;

    private EntityManagerFactory emf = null;

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

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

    public void create(Produto produto) {
        EntityManager em = getEntityManager();
        try {
            em.getTransaction().begin();
            em.persist(produto);
            em.getTransaction().commit();
        } finally {
            em.close();
        }
    }

    public void edit(Produto produto) throws
NonexistentEntityException, Exception {
        EntityManager em = getEntityManager();
```

```
        try {
            em.getTransaction().begin();
            produto = em.merge(produto);
            em.getTransaction().commit();
        } catch (Exception ex) {
            Integer id = produto.getIdProduto();
            if (findProduto(id) == null) {
                throw new NonexistentEntityException("The produto with
id " + id + " no longer exists.");
            }
            throw ex;
        } finally {
            em.close();
        }
    }

    public void destroy(Integer id) throws NonexistentEntityException {
        EntityManager em = getEntityManager();
        try {
            em.getTransaction().begin();
            Produto produto;
            try {
                produto = em.getReference(Produto.class, id);
                produto.getIdProduto();
            } catch (EntityNotFoundException enfe) {
                throw new NonexistentEntityException("The produto with
id " + id + " no longer exists.", enfe);
            }
            em.remove(produto);
            em.getTransaction().commit();
        } finally {
            em.close();
        }
    }

    public List<Produto> findProdutoEntities() {
        return findProdutoEntities(true, -1, -1);
    }

    public List<Produto> findProdutoEntities(int maxResults, int
```

```
firstResult) {
        return findProdutoEntities(false, maxResults, firstResult);
    }

    private List<Produto> findProdutoEntities(boolean all, int
maxResults, int firstResult) {
        EntityManager em = getEntityManager();
        try {
            CriteriaQuery cq = em.getCriteriaBuilder().createQuery();
            cq.select(cq.from(Produto.class));
            Query q = em.createQuery(cq);
            if (!all) {
                q.setMaxResults(maxResults);
                q.setFirstResult(firstResult);
            }
            return q.getResultList();
        } finally {
            em.close();
        }
    }

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

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

```
            em.close();
        }
    }
}
```

7.1.3.1. Arquivo "UsuarioJpaController.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 controller;

import controller.exceptions.NonexistentEntityException;
import model.Usuario;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Query;
import javax.persistence.EntityNotFoundException;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;
import java.io.Serializable;
import java.util.List;

/**
 *
 * @author Ivan
 */
public class UsuarioJpaController implements Serializable {

    private static final long serialVersionUID = 1L;

    private EntityManagerFactory emf = null;

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

```
    }

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

    public void create(Usuario usuario) {
        EntityManager em = getEntityManager();
        try {
            em.getTransaction().begin();
            em.persist(usuario);
            em.getTransaction().commit();
        } finally {
            em.close();
        }
    }

    public void edit(Usuario usuario) throws
NonexistentEntityException, Exception {
        EntityManager em = getEntityManager();
        try {
            em.getTransaction().begin();
            usuario = em.merge(usuario);
            em.getTransaction().commit();
        } catch (Exception ex) {
            Integer id = usuario.getIdUsuario();
            if (findUsuario(id) == null) {
                throw new NonexistentEntityException("The usuario with
id " + id + " no longer exists.");
            }
            throw ex;
        } finally {
            em.close();
        }
    }

    public void destroy(Integer id) throws NonexistentEntityException {
        EntityManager em = getEntityManager();
        try {
```

```
            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);
            }
            em.remove(usuario);
            em.getTransaction().commit();
        } finally {
            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();
        }
    }

    // Método especial pedido no roteiro:
    public Usuario findUsuario(String login, String senha) {
        EntityManager em = getEntityManager();
        try {
            Query q = em.createQuery("SELECT u FROM Usuario u WHERE
u.login = :login AND u.senha = :senha");
            q.setParameter("login", login);
            q.setParameter("senha", senha);
            List<Usuario> results = q.getResultList();
            if (results.isEmpty()) {
                return null;
            } else {
                return results.get(0);
            }
        } finally {
```

```
            em.close();
        }
    }
}
```

7.1.4. Arquivos do pacote Java “controller.exceptions” do projeto “CadastroServer”

7.1.4.1. Arquivo “IllegalOrphanException.java”

```
package controller.exceptions;

import java.util.ArrayList;
import java.util.List;

public class IllegalOrphanException extends Exception {
    private List<String> messages;
    public IllegalOrphanException(List<String> messages) {
        super((messages != null && messages.size() > 0 ?
messages.get(0) : null));
        if (messages == null) {
            this.messages = new ArrayList<String>();
        }
        else {
            this.messages = messages;
        }
    }
    public List<String> getMessages() {
        return messages;
    }
}
```

7.1.4.2. Arquivo “NonexistentEntityException.java”

```
package controller.exceptions;

public class NonexistentEntityException extends Exception {
    public NonexistentEntityException(String message, Throwable cause)
{
```

```
        super(message, cause);
    }
    public NonexistentEntityException(String message) {
        super(message);
    }
}
```

7.1.4.3. Arquivo "PreexistingEntityException.java"

```
package controller.exceptions;

public class PreexistingEntityException extends Exception {
    public PreexistingEntityException(String message, Throwable cause)
{
        super(message, cause);
    }
    public PreexistingEntityException(String message) {
        super(message);
    }
}
```

7.1.5. Arquivos do pacote Java "model" do projeto "CadastroServer"

7.1.5.1. Arquivo "Movimento.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 model;

import java.io.Serializable;
import java.math.BigDecimal;
import javax.persistence.Basic;
import javax.persistence.Column;
```

```
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.ManyToOne;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;

/**
 *
 * @author Ivan
 */
@Entity
@Table(name = "Movimento")
@NamedQueries({
    @NamedQuery(name = "Movimento.findAll", query = "SELECT m FROM
Movimento m"),
    @NamedQuery(name = "Movimento.findByIdMovimento", query = "SELECT m
FROM Movimento m WHERE m.idMovimento = :idMovimento"),
    @NamedQuery(name = "Movimento.findByQuantidade", query = "SELECT m
FROM Movimento m WHERE m.quantidade = :quantidade"),
    @NamedQuery(name = "Movimento.findByTipo", query = "SELECT m FROM
Movimento m WHERE m.tipo = :tipo"),
    @NamedQuery(name = "Movimento.findByValorUnitario", query = "SELECT
m FROM Movimento m WHERE m.valorUnitario = :valorUnitario")})
public class Movimento implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "idMovimento")
    private Integer idMovimento;
    @Basic(optional = false)
    @Column(name = "quantidade")
    private int quantidade;
    @Column(name = "tipo")
    private Character tipo;
```

```
    // @Max(value=?)  @Min(value=?)//if you know range of your decimal
fields consider using these annotations to enforce field validation
    @Basic(optional = false)
    @Column(name = "valorUnitario")
    private BigDecimal valorUnitario;
    @JoinColumn(name = "idPessoa", referencedColumnName = "idPessoa")
    @ManyToOne(optional = false)
    private Pessoa idPessoa;
    @JoinColumn(name = "idProduto", referencedColumnName = "idProduto")
    @ManyToOne(optional = false)
    private Produto idProduto;
    @JoinColumn(name = "idUsuario", referencedColumnName = "idUsuario")
    @ManyToOne(optional = false)
    private Usuario idUsuario;

    public Movimento() {
    }

    public Movimento(Integer idMovimento) {
        this.idMovimento = idMovimento;
    }

    public Movimento(Integer idMovimento, int quantidade, BigDecimal
valorUnitario) {
        this.idMovimento = idMovimento;
        this.quantidade = quantidade;
        this.valorUnitario = valorUnitario;
    }

    public Integer getIdMovimento() {
        return idMovimento;
    }

    public void setIdMovimento(Integer idMovimento) {
        this.idMovimento = idMovimento;
    }

    public int getQuantidade() {
        return quantidade;
```

```
    }

    public void setQuantidade(int quantidade) {
        this.quantidade = quantidade;
    }

    public Character getTipo() {
        return tipo;
    }

    public void setTipo(Character tipo) {
        this.tipo = tipo;
    }

    public BigDecimal getValorUnitario() {
        return valorUnitario;
    }

    public void setValorUnitario(BigDecimal valorUnitario) {
        this.valorUnitario = valorUnitario;
    }

    public Pessoa getIdPessoa() {
        return idPessoa;
    }

    public void setIdPessoa(Pessoa idPessoa) {
        this.idPessoa = idPessoa;
    }

    public Produto getIdProduto() {
        return idProduto;
    }

    public void setIdProduto(Produto idProduto) {
        this.idProduto = idProduto;
    }

    public Usuario getIdUsuario() {
```

```
        return idUsuario;
    }

    public void setIdUsuario(Usuario idUsuario) {
        this.idUsuario = idUsuario;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idMovimento != null ? idMovimento.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id
fields are not set
        if (!(object instanceof Movimento)) {
            return false;
        }
        Movimento other = (Movimento) object;
        if ((this.idMovimento == null && other.idMovimento != null) ||
(this.idMovimento != null &&
this.idMovimento.equals(other.idMovimento))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "model.Movimento[ idMovimento=" + idMovimento + " ]";
    }

}
```

7.1.5.2. Arquivo "Pessoa.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 model;

import java.io.Serializable;
import java.util.Collection;
import javax.persistence.Basic;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.OneToMany;
import javax.persistence.OneToOne;
import javax.persistence.Table;

/**
 *
 * @author Ivan
 */
@Entity
@Table(name = "Pessoa")
@NamedQueries({
    @NamedQuery(name = "Pessoa.findAll", query = "SELECT p FROM Pessoa
p"),
    @NamedQuery(name = "Pessoa.findByIdPessoa", query = "SELECT p FROM
Pessoa p WHERE p.idPessoa = :idPessoa"),
    @NamedQuery(name = "Pessoa.findByNome", query = "SELECT p FROM
Pessoa p WHERE p.nome = :nome"),
    @NamedQuery(name = "Pessoa.findByLogradouro", query = "SELECT p
FROM Pessoa p WHERE p.logradouro = :logradouro"),
    @NamedQuery(name = "Pessoa.findByCidade", query = "SELECT p FROM
Pessoa p WHERE p.cidade = :cidade"),
    @NamedQuery(name = "Pessoa.findByEstado", query = "SELECT p FROM
Pessoa p WHERE p.estado = :estado"),
    @NamedQuery(name = "Pessoa.findByTelefone", query = "SELECT p FROM
```

```
Pessoa p WHERE p.telefone = :telefone"),
    @NamedQuery(name = "Pessoa.findByEmail", query = "SELECT p FROM
Pessoa p WHERE p.email = :email")})
public class Pessoa implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @Column(name = "idPessoa")
    private Integer idPessoa;
    @Basic(optional = false)
    @Column(name = "nome")
    private String nome;
    @Basic(optional = false)
    @Column(name = "logradouro")
    private String logradouro;
    @Basic(optional = false)
    @Column(name = "cidade")
    private String cidade;
    @Basic(optional = false)
    @Column(name = "estado")
    private String estado;
    @Basic(optional = false)
    @Column(name = "telefone")
    private String telefone;
    @Basic(optional = false)
    @Column(name = "email")
    private String email;
    @OneToOne(cascade = CascadeType.ALL, mappedBy = "pessoa")
    private PessoaJuridica pessoaJuridica;
    @OneToOne(cascade = CascadeType.ALL, mappedBy = "pessoa")
    private PessoaFisica pessoaFisica;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "idPessoa")
    private Collection<Movimento> movimentoCollection;

    public Pessoa() {
    }

    public Pessoa(Integer idPessoa) {
```

```
        this.idPessoa = idPessoa;
    }

    public Pessoa(Integer idPessoa, String nome, String logradouro,
String cidade, String estado, String telefone, String email) {
        this.idPessoa = idPessoa;
        this.nome = nome;
        this.logradouro = logradouro;
        this.cidade = cidade;
        this.estado = estado;
        this.telefone = telefone;
        this.email = email;
    }

    public Integer getIdPessoa() {
        return idPessoa;
    }

    public void setIdPessoa(Integer idPessoa) {
        this.idPessoa = idPessoa;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public String getLogradouro() {
        return logradouro;
    }

    public void setLogradouro(String logradouro) {
        this.logradouro = logradouro;
    }

    public String getCidade() {
```

```
        return cidade;
    }

    public void setCidade(String cidade) {
        this.cidade = cidade;
    }

    public String getEstado() {
        return estado;
    }

    public void setEstado(String estado) {
        this.estado = estado;
    }

    public String getTelefone() {
        return telefone;
    }

    public void setTelefone(String telefone) {
        this.telefone = telefone;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    public PessoaJuridica getPessoaJuridica() {
        return pessoaJuridica;
    }

    public void setPessoaJuridica(PessoaJuridica pessoaJuridica) {
        this.pessoaJuridica = pessoaJuridica;
    }
```

```
    public PessoaFisica getPessoaFisica() {
        return pessoaFisica;
    }

    public void setPessoaFisica(PessoaFisica pessoaFisica) {
        this.pessoaFisica = pessoaFisica;
    }

    public Collection<Movimento> getMovimentoCollection() {
        return movimentoCollection;
    }

    public void setMovimentoCollection(Collection<Movimento>
movimentoCollection) {
        this.movimentoCollection = movimentoCollection;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idPessoa != null ? idPessoa.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id
fields are not set
        if (!(object instanceof Pessoa)) {
            return false;
        }
        Pessoa other = (Pessoa) object;
        if ((this.idPessoa == null && other.idPessoa != null) ||
(this.idPessoa != null && !this.idPessoa.equals(other.idPessoa))) {
            return false;
        }
        return true;
    }

    @Override
```

```
    public String toString() {
        return "model.Pessoa[ idPessoa=" + idPessoa + " ]";
    }

}
```

7.1.5.3. Arquivo "PessoaFisica.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 model;

import java.io.Serializable;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.OneToOne;
import javax.persistence.Table;

/**
 *
 * @author Ivan
 */
@Entity
@Table(name = "Pessoa_Fisica")
@NamedQueries({
    @NamedQuery(name = "PessoaFisica.findAll", query = "SELECT p FROM
PessoaFisica p"),
    @NamedQuery(name = "PessoaFisica.findByIdPessoa", query = "SELECT p
FROM PessoaFisica p WHERE p.idPessoa = :idPessoa"),
    @NamedQuery(name = "PessoaFisica.findByCpf", query = "SELECT p FROM
PessoaFisica p WHERE p.cpf = :cpf")})
```

```
public class PessoaFisica implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @Basic(optional = false)
    @Column(name = "idPessoa")
    private Integer idPessoa;
    @Basic(optional = false)
    @Column(name = "cpf")
    private String cpf;
    @JoinColumn(name = "idPessoa", referencedColumnName = "idPessoa",
insertable = false, updatable = false)
    @OneToOne(optional = false)
    private Pessoa pessoa;

    public PessoaFisica() {
    }

    public PessoaFisica(Integer idPessoa) {
        this.idPessoa = idPessoa;
    }

    public PessoaFisica(Integer idPessoa, String cpf) {
        this.idPessoa = idPessoa;
        this.cpf = cpf;
    }

    public Integer getIdPessoa() {
        return idPessoa;
    }

    public void setIdPessoa(Integer idPessoa) {
        this.idPessoa = idPessoa;
    }

    public String getCpf() {
        return cpf;
    }
```

```
    public void setCpf(String cpf) {
        this.cpf = cpf;
    }

    public Pessoa getPessoa() {
        return pessoa;
    }

    public void setPessoa(Pessoa pessoa) {
        this.pessoa = pessoa;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idPessoa != null ? idPessoa.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id
fields are not set
        if (!(object instanceof PessoaFisica)) {
            return false;
        }
        PessoaFisica other = (PessoaFisica) object;
        if ((this.idPessoa == null && other.idPessoa != null) ||
(this.idPessoa != null && !this.idPessoa.equals(other.idPessoa))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "model.PessoaFisica[ idPessoa=" + idPessoa + " ]";
    }
```

```
}
```

7.1.5.4. Arquivo "PessoaJuridica.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 model;

import java.io.Serializable;
import javax.persistence.Basic;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.OneToOne;
import javax.persistence.Table;

/**
 *
 * @author Ivan
 */
@Entity
@Table(name = "Pessoa_Juridica")
@NamedQueries({
    @NamedQuery(name = "PessoaJuridica.findAll", query = "SELECT p FROM
PessoaJuridica p"),
    @NamedQuery(name = "PessoaJuridica.findByIdPessoa", query = "SELECT
p FROM PessoaJuridica p WHERE p.idPessoa = :idPessoa"),
    @NamedQuery(name = "PessoaJuridica.findByCnpj", query = "SELECT p
FROM PessoaJuridica p WHERE p.cnpj = :cnpj")})
public class PessoaJuridica implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
```

```
    @Basic(optional = false)
    @Column(name = "idPessoa")
    private Integer idPessoa;
    @Basic(optional = false)
    @Column(name = "cnpj")
    private String cnpj;
    @JoinColumn(name = "idPessoa", referencedColumnName = "idPessoa",
insertable = false, updatable = false)
    @OneToOne(optional = false)
    private Pessoa pessoa;

    public PessoaJuridica() {
    }

    public PessoaJuridica(Integer idPessoa) {
        this.idPessoa = idPessoa;
    }

    public PessoaJuridica(Integer idPessoa, String cnpj) {
        this.idPessoa = idPessoa;
        this.cnpj = cnpj;
    }

    public Integer getIdPessoa() {
        return idPessoa;
    }

    public void setIdPessoa(Integer idPessoa) {
        this.idPessoa = idPessoa;
    }

    public String getCnpj() {
        return cnpj;
    }

    public void setCnpj(String cnpj) {
        this.cnpj = cnpj;
    }
```

```
    public Pessoa getPessoa() {
        return pessoa;
    }

    public void setPessoa(Pessoa pessoa) {
        this.pessoa = pessoa;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idPessoa != null ? idPessoa.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id
fields are not set
        if (!(object instanceof PessoaJuridica)) {
            return false;
        }
        PessoaJuridica other = (PessoaJuridica) object;
        if ((this.idPessoa == null && other.idPessoa != null) ||
(this.idPessoa != null && !this.idPessoa.equals(other.idPessoa))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "model.PessoaJuridica[ idPessoa=" + idPessoa + " ]";
    }

}
```

7.1.5.5. Arquivo "Produto.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 model;

import java.io.Serializable;
import java.math.BigDecimal;
import java.util.Collection;
import javax.persistence.Basic;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.OneToMany;
import javax.persistence.Table;

/**
 *
 * @author Ivan
 */
@Entity
@Table(name = "Produto")
@NamedQueries({
    @NamedQuery(name = "Produto.findAll", query = "SELECT p FROM
Produto p"),
    @NamedQuery(name = "Produto.findByIdProduto", query = "SELECT p
FROM Produto p WHERE p.idProduto = :idProduto"),
    @NamedQuery(name = "Produto.findByNome", query = "SELECT p FROM
Produto p WHERE p.nome = :nome"),
    @NamedQuery(name = "Produto.findByQuantidade", query = "SELECT p
FROM Produto p WHERE p.quantidade = :quantidade"),
    @NamedQuery(name = "Produto.findByPrecoVenda", query = "SELECT p
FROM Produto p WHERE p.precoVenda = :precoVenda")})
```

```
public class Produto implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "idProduto")
    private Integer idProduto;
    @Basic(optional = false)
    @Column(name = "nome")
    private String nome;
    @Basic(optional = false)
    @Column(name = "quantidade")
    private int quantidade;
    // @Max(value=?)  @Min(value=?)//if you know range of your decimal
fields consider using these annotations to enforce field validation
    @Basic(optional = false)
    @Column(name = "precoVenda")
    private BigDecimal precoVenda;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "idProduto")
    private Collection<Movimento> movimentoCollection;

    public Produto() {
    }

    public Produto(Integer idProduto) {
        this.idProduto = idProduto;
    }

    public Produto(Integer idProduto, String nome, int quantidade,
BigDecimal precoVenda) {
        this.idProduto = idProduto;
        this.nome = nome;
        this.quantidade = quantidade;
        this.precoVenda = precoVenda;
    }

    public Integer getIdProduto() {
        return idProduto;
```

```
    }

    public void setIdProduto(Integer idProduto) {
        this.idProduto = idProduto;
    }

    public String getNome() {
        return nome;
    }

    public void setNome(String nome) {
        this.nome = nome;
    }

    public int getQuantidade() {
        return quantidade;
    }

    public void setQuantidade(int quantidade) {
        this.quantidade = quantidade;
    }

    public BigDecimal getPrecoVenda() {
        return precoVenda;
    }

    public void setPrecoVenda(BigDecimal precoVenda) {
        this.precoVenda = precoVenda;
    }

    public Collection<Movimento> getMovimentoCollection() {
        return movimentoCollection;
    }

    public void setMovimentoCollection(Collection<Movimento>
movimentoCollection) {
        this.movimentoCollection = movimentoCollection;
    }
```

```
    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idProduto != null ? idProduto.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id
fields are not set
        if (!(object instanceof Produto)) {
            return false;
        }
        Produto other = (Produto) object;
        if ((this.idProduto == null && other.idProduto != null) ||
(this.idProduto != null && !this.idProduto.equals(other.idProduto))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "model.Produto[ idProduto=" + idProduto + " ]";
    }

}
```

7.1.4.6. Arquivo "Usuario.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 model;

import java.io.Serializable;
```

```
import java.util.Collection;
import javax.persistence.Basic;
import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.OneToMany;
import javax.persistence.Table;

/**
 *
 * @author Ivan
 */
@Entity
@Table(name = "Usuario")
@NamedQueries({
    @NamedQuery(name = "Usuario.findAll", query = "SELECT u FROM
Usuario u"),
    @NamedQuery(name = "Usuario.findByIdUsuario", query = "SELECT u
FROM Usuario u WHERE u.idUsuario = :idUsuario"),
    @NamedQuery(name = "Usuario.findByLogin", query = "SELECT u FROM
Usuario u WHERE u.login = :login"),
    @NamedQuery(name = "Usuario.findBySenha", query = "SELECT u FROM
Usuario u WHERE u.senha = :senha")})
public class Usuario implements Serializable {

    private static final long serialVersionUID = 1L;
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Basic(optional = false)
    @Column(name = "idUsuario")
    private Integer idUsuario;
    @Basic(optional = false)
    @Column(name = "login")
    private String login;
    @Basic(optional = false)
```

```
    @Column(name = "senha")
    private String senha;
    @OneToMany(cascade = CascadeType.ALL, mappedBy = "idUsuario")
    private Collection<Movimento> movimentoCollection;

    public Usuario() {
    }

    public Usuario(Integer idUsuario) {
        this.idUsuario = idUsuario;
    }

    public Usuario(Integer idUsuario, String login, String senha) {
        this.idUsuario = idUsuario;
        this.login = login;
        this.senha = senha;
    }

    public Integer getIdUsuario() {
        return idUsuario;
    }

    public void setIdUsuario(Integer idUsuario) {
        this.idUsuario = idUsuario;
    }

    public String getLogin() {
        return login;
    }

    public void setLogin(String login) {
        this.login = login;
    }

    public String getSenha() {
        return senha;
    }

    public void setSenha(String senha) {
```

```
        this.senha = senha;
    }

    public Collection<Movimento> getMovimentoCollection() {
        return movimentoCollection;
    }

    public void setMovimentoCollection(Collection<Movimento>
movimentoCollection) {
        this.movimentoCollection = movimentoCollection;
    }

    @Override
    public int hashCode() {
        int hash = 0;
        hash += (idUsuario != null ? idUsuario.hashCode() : 0);
        return hash;
    }

    @Override
    public boolean equals(Object object) {
        // TODO: Warning - this method won't work in the case the id
fields are not set
        if (!(object instanceof Usuario)) {
            return false;
        }
        Usuario other = (Usuario) object;
        if ((this.idUsuario == null && other.idUsuario != null) ||
(this.idUsuario != null && !this.idUsuario.equals(other.idUsuario))) {
            return false;
        }
        return true;
    }

    @Override
    public String toString() {
        return "model.Usuario[ idUsuario=" + idUsuario + " ]";
    }

}
```

7.2. Códigos fonte do projeto de aplicativo Java (with Ant) “CadastroClient”

7.2.1. Arquivos do pacote Java “cadastroclient” do projeto “CadastroClient”

7.1.5.2. Arquivo “CadastroClientMain.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 model.Produto;

import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;

/**
 *
 * @author Ivan
 */
public class CadastroClientMain {

    private static final Logger LOGGER =
Logger.getLogger(CadastroClientMain.class.getName());
```

```
    /**
     * @param args the command line arguments
     */
    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());
        ) {

            // (1) Se chegamos aqui, a conexão com o servidor na porta
4321 foi estabelecida,
            //     e os canais de entrada e saída foram abertos com
sucesso.

            // (2) Enviar login e senha
            String login = "op1";
            String senha = "op1";
            out.writeObject(login);
            out.writeObject(senha);
            out.flush();

            // (3) Se chegamos aqui, usuário foi autenticado com
sucesso
            System.out.println("Usuario conectado com sucesso");

            // (4) Enviar comando "L"
            String comando = "L";
            out.writeObject(comando);
            out.flush();

            // (5) Receber lista de produtos
            List<Produto> produtos = (List<Produto>) in.readObject();

            // (6) Exibir nomes dos produtos
            for (Produto p : produtos) {
                System.out.println(p.getNome());
```

```
            }
        } catch (Exception e) {
            LOGGER.log(Level.SEVERE, "Erro no cliente", e);
        }
    }
}
```

7.2.2. Arquivos do pacote Java “model” do projeto “CadastroClient”

Este pacote Java (i.e., “model” do projeto “CadastroClient”) contém os seguintes arquivos:

1. Movimento.java
2. Pessoa.java
3. PessoaFisica.java
4. PessoaJuridica.java
5. Produto.java
6. Usuario.java

O conteúdo desses arquivos é o mesmo conteúdo dos arquivos de mesmo nome localizados no pacote Java de mesmo nome (i.e., “model”) contido no projeto “CadastroServer”. A caminho preciso da pasta que desse pacote “model” é “/CadastroSocketServerClient_MP_N5_M3/CadastroClient/src/model” no repositório do GitHub onde se localiza os três projetos (i.e., “CadastroServer”, “CadastroClient” e “CadastroClientV2”) da Missão Prática | Nível 5 | Mundo 3 (i.e., o repositório “https://github.com/ivanmortimer/CadastroSocketServerClient_MP_N5_M3”).