



Estácio



Relatório discente de acompanhamento

Universidade	Estácio de Sá
Campus	Campo Grande - Cariacica / ES
Nome do Curso	Desenvolvimento Full Stack
Nome da Disciplina	RPG0018 - Por que não paralelizar
Turma	9001
Semestre	2024.3
Integrante	Antônio Vitor Serra dos Santos
Matrícula	2023 0701 4834

Servidores e clientes baseados em Socket, com uso de Threads tanto no lado cliente quanto no lado servidor, acessando o banco de dados via JPA.

✂ Objetivos da prática:

- Criar servidores Java com base em Sockets.
- Criar clientes síncronos para servidores com base em Sockets.
- Criar clientes assíncronos para servidores com base em Sockets.
- Utilizar Threads para implementação de processos paralelos.
- No final do exercício, o aluno terá criado um servidor Java baseado em Socket, com acesso ao banco de dados via JPA, além de utilizar os recursos nativos do Java para implementação de clientes síncronos e assíncronos. As Threads serão usadas tanto no servidor, para viabilizar múltiplos clientes paralelos, quanto no cliente, para implementar a resposta assíncrona.

Repositório GitHub: <http://github.com/t8ninho/Mundo-3-Nivel-5/>

Cariacica / ES
2024

1º Procedimento | Criando o Servidor e Cliente de Teste

* Código solicitados:

CadastroServer.java

```
package cadastroserver;

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

/**
 *
 * @author tonis
 */

public class CadastroServer {

    public static void main(String[] args) {

        EntityManagerFactory emf = Persistence.createEntityManagerFactory("lojaPU");
        ProdutoJpaController produtoCtrl = new ProdutoJpaController(emf);
        UsuarioJpaController usuarioCtrl = new UsuarioJpaController(emf);

        try (ServerSocket serverSocket = new ServerSocket(4321)) {
            System.out.println("Servidor iniciado na porta 4321.");
            while (true) {
                Socket clientSocket = serverSocket.accept();
                new Thread(new CadastroThread(produtoCtrl, usuarioCtrl, clientSocket)).start();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

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 java.io.EOFException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.util.List;
import controller.ProdutoJpaController;
import controller.UsuarioJpaController;
import model.Produto;
import model.Usuario;

/**
 *
 * @author tonis
 */
public class CadastroThread implements Runnable {

    private ProdutoJpaController ctrl;
    private UsuarioJpaController ctrlUsu;
    private 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())) {

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

            Usuario usuario = ctrlUsu.findUsuario(login, senha);
            if (usuario == null) {
                out.writeObject("Credenciais invalidas. Conexao encerrada.");
                s1.close();
                return;
            }

            out.writeObject("Usuario conectado com sucesso.");
            String comando;
            while ((comando = (String) in.readObject()) != null) {
                if (comando.equalsIgnoreCase("L")) {
                    List<Produto> produtos = ctrl.findProdutoEntities();
                    out.writeObject(produtos);
                } else {
                    out.writeObject("Comando desconhecido.");
                }
            }
        } catch (EOFException eof) {
            System.out.println("O cliente fechou a conexao.");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

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 javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.NoResultException;
import javax.persistence.Query;
import model.Usuario;

/**
 *
 * @author tonis
 */
public class UsuarioJpaController {

    private EntityManagerFactory emf;

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

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

    public Usuario findUsuario(String login, String senha) {
        EntityManager em = null;
        try {
            em = getEntityManager();
            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);
            return (Usuario) query.getSingleResult();
        } catch (NoResultException e) {
            return null;
        } finally {
            if (em != null) {
                em.close();
            }
        }
    }
}
```

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 java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Query;
import javax.persistence.criteria.CriteriaQuery;
import model.Produto;

/**
 *
 * @author tonis
 */
public class ProdutoJpaController {

    private EntityManagerFactory emf;

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

    public List<Produto> findProdutoEntities() {
        EntityManager em = getEntityManager();
        try {
            CriteriaQuery<Produto> cq = em.getCriteriaBuilder().createQuery(Produto.class);
            cq.select(cq.from(Produto.class));
            Query q = em.createQuery(cq);
            return q.getResultList();
        } finally {
            em.close();
        }
    }

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

    public void edit(Produto produto) {
        EntityManager em = null;
        try {
            em = getEntityManager();
            em.getTransaction().begin();
            produto = em.merge(produto);
            em.getTransaction().commit();
        } catch (Exception ex) {
            if (em != null && em.getTransaction().isActive()) {
                em.getTransaction().rollback();
            }
            ex.printStackTrace();
        } finally {
            if (em != null) {
                em.close();
            }
        }
    }
}
```

```

        em.close();
    }
}

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

```

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

/**
 *
 * @author tonis
 */
public class CadastroClient {
    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())) {

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

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

            out.writeObject("L");

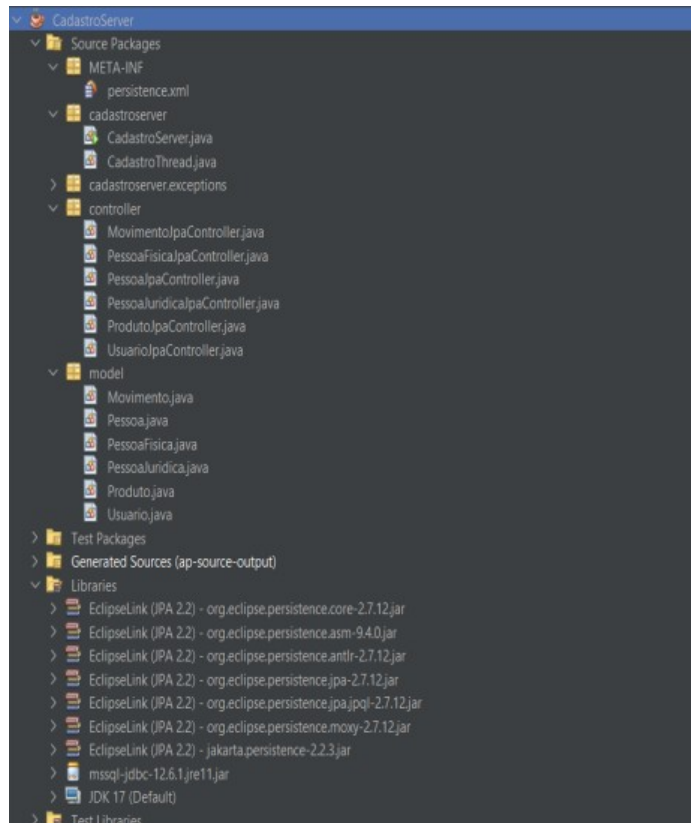
            List<Produto> produtos = (List<Produto>) in.readObject();
            for (Produto p : produtos) {
                System.out.println(p.getNome());
            }

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

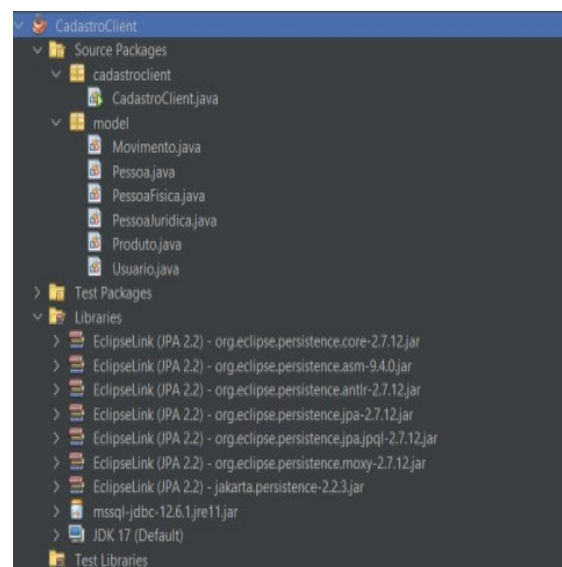
```

* Resultados:

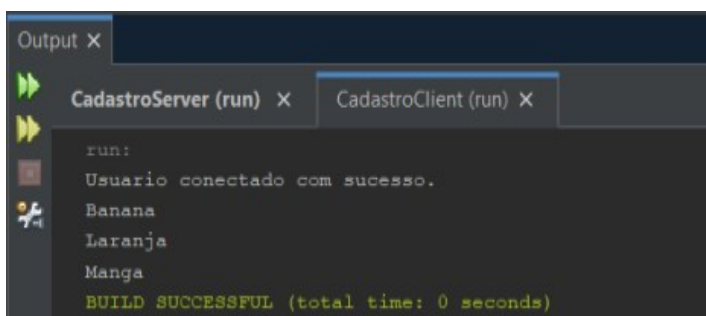
Estrutura do projeto CadastroServer:



Estrutura do projeto CadastroClient:



Resultado da execução:



*** Análise e Conclusão:**

a) Como funcionam as classes Socket e ServerSocket?

As classes Socket e ServerSocket são usadas para comunicação em rede. O ServerSocket fica escutando e aceita conexões, enquanto o Socket inicia e gerencia a conexão com o servidor.

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

As portas são usadas para identificar serviços em um computador. Cada aplicação tem uma porta específica, permitindo que o servidor encaminhe as solicitações para o serviço correto.

c) Para que servem as classes de entrada e saída InputStream e OutputStream, e por que os objetos transmitidos devem ser serializáveis?

As classes InputStream e OutputStream permitem ler e escrever objetos em fluxos de dados. Para transmissão pela rede, os objetos precisam ser serializáveis.

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

As classes de entidades JPA no cliente isolam o acesso ao banco de dados ao encapsular a lógica de manipulação de dados e oferecer métodos seguros para operações. O uso de transações assegura consistência e isolamento nas operações realizadas.

2º Procedimento | Servidor Completo e Cliente Assíncrono

* Código solicitados:

CadastroThreadV2.java

```
package cadastroserver;

import controller.MovimentoJpaController;
import controller.PessoaJpaController;
import controller.ProdutoJpaController;
import controller.UsuarioJpaController;
import java.io.IOException;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.math.BigDecimal;
import java.net.Socket;
import java.util.List;
import java.util.logging.Level;
import java.util.logging.Logger;
import model.Movimento;
import model.Produto;
import model.Usuario;

/**
 *
 * @author tonis
 */
public class CadastroThreadV2 implements Runnable {

    private final ProdutoJpaController ctrlProd;
    private final UsuarioJpaController ctrlUsu;
    private final MovimentoJpaController ctrlMov;
    private final PessoaJpaController ctrlPessoa;
    private final Socket socket;
    private ObjectOutputStream out;
    private ObjectInputStream in;

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

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

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

            Usuario usuario = ctrlUsu.findUsuario(login, senha);
            if (usuario == null) {
                out.writeObject("Credenciais inválidas. Conexão encerrada.");
                return;
            }
        }
    }
}
```

```

out.writeObject("Usuário conectado com sucesso.");

String comando;
while ((comando = (String) in.readObject()) != null) {
    if (comando.equalsIgnoreCase("X")) {
        out.writeObject("Conexão encerrada pelo cliente.");
        System.out.println("Cliente solicitou encerramento da conexão.");
        break;
    }

    switch (comando.toUpperCase()) {
        case "L":
            List<Produto> produtos = ctrlProd.findProdutoEntities();
            out.writeObject(produtos);
            break;
        case "E":
        case "S":
            Integer idPessoa = (Integer) in.readObject();
            Integer idProduto = (Integer) in.readObject();
            Integer quantidade = (Integer) in.readObject();
            BigDecimal valorUnitario = (BigDecimal) in.readObject();

            Movimento movimento = new Movimento();
            movimento.setIdUsuario(usuario);
            movimento.setTipo(comando.toUpperCase().charAt(0));
            movimento.setIdPessoa(ctrlPessoa.findPessoa(idPessoa));
            movimento.setIdProduto(ctrlProd.findProduto(idProduto));
            movimento.setQuantidade(quantidade);
            movimento.setValorUnitario(valorUnitario);

            ctrlMov.create(movimento);

            Produto produto = ctrlProd.findProduto(idProduto);
            if (comando.equalsIgnoreCase("E")) {
                produto.setQuantidade(produto.getQuantidade() + quantidade);
            } else {
                produto.setQuantidade(produto.getQuantidade() - quantidade);
            }
            ctrlProd.edit(produto);

            out.writeObject("Movimento registrado com sucesso.");
            break;
        default:
            out.writeObject("Comando desconhecido.");
            break;
    }
}
} catch (IOException | ClassNotFoundException ex) {
    Logger.getLogger(CadastroThreadV2.class.getName()).log(Level.SEVERE, null, ex);
} finally {
    try {
        if (out != null) {
            out.close();
        }
        if (in != null) {
            in.close();
        }
        socket.close();
    } catch (IOException ex) {
        Logger.getLogger(CadastroThreadV2.class.getName()).log(Level.SEVERE, null, ex);
    }
}
}
}
}
}

```

CadastroClientV2.java

```
package cadastroclientv2;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.math.BigDecimal;
import java.net.Socket;
import java.util.List;
import javax.swing.JFrame;
import javax.swing.JTextArea;
import javax.swing.SwingUtilities;
import model.Produto;

/**
 *
 * @author tonis
 */

public class CadastroClientV2 {

    public static void main(String[] args) {
        try {
            Socket socket = new Socket("localhost", 4321);

            ObjectOutputStream out = new ObjectOutputStream(socket.getOutputStream());
            ObjectInputStream in = new ObjectInputStream(socket.getInputStream());

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

            System.out.print("Digite o login: ");
            String login = reader.readLine();
            System.out.print("Digite a senha: ");
            String senha = reader.readLine();

            out.writeObject(login);
            out.writeObject(senha);

            JFrame frame = new JFrame("Mensagens do Servidor");
            JTextArea textArea = new JTextArea(20, 50);
            textArea.setEditable(false);
            frame.add(textArea);
            frame.pack();
            frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            SwingUtilities.invokeLater(() -> frame.setVisible(true));

            new Thread(() -> {
                try {
                    while (true) {
                        Object response = in.readObject();
                        if (response instanceof String) {
                            String message = (String) response;
                            if ("Conexão encerrada pelo cliente.".equals(message)) {
                                break;
                            }
                        }
                        SwingUtilities.invokeLater(() -> textArea.append(message + "\n"));
                    } else if (response instanceof List<?>) {
                        List<Produto> produtos = (List<Produto>) response;
                        SwingUtilities.invokeLater(() -> {
                            textArea.append("Produtos:\n");
                        });
                    }
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }) {
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

```

        for (Produto produto : produtos) {
            textArea.append(produto.getNome() + " - Quantidade: "
                + produto.getQuantidade() + "\n");
        }
    });
}
} catch (Exception e) {
    e.printStackTrace();
} finally {
    try {
        if (in != null) {
            in.close();
        }
        if (out != null) {
            out.close();
        }
        if (socket != null) {
            socket.close();
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
}).start();

while (true) {
    System.out.println("Menu:");
    System.out.println("L - Listar");
    System.out.println("E - Entrada");
    System.out.println("S - Saída");
    System.out.println("X - Finalizar");
    String command = reader.readLine();

    out.writeObject(command);

    if (command.equalsIgnoreCase("X")) {
        System.out.println("Finalizando...");
        break;
    }

    if (command.equalsIgnoreCase("L")) {
        continue;
    }

    if (command.equalsIgnoreCase("E") || command.equalsIgnoreCase("S")) {
        System.out.print("Id da pessoa: ");
        int idPessoa = Integer.parseInt(reader.readLine());
        out.writeObject(idPessoa);

        System.out.print("Id do produto: ");
        int idProduto = Integer.parseInt(reader.readLine());
        out.writeObject(idProduto);

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

        System.out.print("Valor unitário: ");
        BigDecimal valorUnitario = new BigDecimal(reader.readLine());
        out.writeObject(valorUnitario);
    }
}
}

```

```

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

```

CadastroServer.java

```

package cadastroserver;

import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import controller.MovimentoJpaController;
import controller.PessoaJpaController;
import controller.ProdutoJpaController;
import controller.UsuarioJpaController;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;

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

    public static void main(String[] args) {

        EntityManagerFactory emf = Persistence.createEntityManagerFactory("lojaPU");

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

        try (ServerSocket serverSocket = new ServerSocket(4321)) {
            while (true) {
                System.out.println("Aguardando conexão...");
                Socket socket = serverSocket.accept();
                System.out.println("Cliente conectado.");

                CadastroThreadV2 thread = new CadastroThreadV2(ctrlProd, ctrlUsu, ctrlMov, ctrlPessoa, socket);
                new Thread(thread).start();
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

ThreadClient.java

```
package cadastroclientv2;

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

/**
 *
 * @author tonis
 */
public class ThreadClient implements Runnable {

    private ObjectInputStream entrada;
    private JTextArea textArea;

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

    @Override
    public void run() {
        try {
            while (true) {
                Object obj = entrada.readObject();
                if (obj instanceof String) {
                    String mensagem = (String) obj;
                    SwingUtilities.invokeLater() -> {
                        textArea.append(mensagem + "\n");
                    };
                } else if (obj instanceof List) {
                    List<Produto> produtos = (List<Produto>) obj;
                    SwingUtilities.invokeLater() -> {
                        for (Produto p : produtos) {
                            textArea.append(p.getNome() + " - " + p.getQuantidade() + "\n");
                        }
                    };
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

SaidaFrame.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 cadastroclientv2;

import javax.swing.JDialog;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;

/**
 *
 * @author tonis
 */
public class SaidaFrame extends JDialog {
    public JTextArea texto;

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

        texto = new JTextArea();
        JScrollPane scrollPane = new JScrollPane(texto);
        scrollPane.setBounds(10, 10, 380, 250);
        add(scrollPane);
    }

    public JTextArea getTextArea() {
        return texto;
    }
}
```

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 java.math.BigDecimal;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Query;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;
import model.Movimento;

/**
 *
 * @author tonis
 */
public class MovimentoJpaController {

    private EntityManagerFactory emf;

    public MovimentoJpaController(EntityManagerFactory emf) {
```



```

        this.emf = emf;
    }

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

    public void create(Movimento movimento) {
        EntityManager em = null;
        try {
            em = getEntityManager();
            em.getTransaction().begin();
            em.persist(movimento);
            em.getTransaction().commit();
        } catch (Exception e) {
            throw new RuntimeException("Erro ao criar movimento.", e);
        } finally {
            if (em != null) {
                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();
    }
}
}
}

```

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 javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.EntityNotFoundException;
import javax.persistence.Query;
import javax.persistence.criteria.CriteriaQuery;
import javax.persistence.criteria.Root;
import model.Pessoa;

/**
 *
 * @author tonis
 */
public class PessoaJpaController {

    private final EntityManagerFactory emf;

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

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

    public void create(Pessoa pessoa) {
        EntityManager em = null;
        try {
            em = getEntityManager();
            em.getTransaction().begin();
            em.persist(pessoa);
            em.getTransaction().commit();
        } catch (Exception e) {
            throw new RuntimeException("Ocorreu um erro ao criar a pessoa.", e);
        } finally {
            if (em != null) {
                em.close();
            }
        }
    }

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

```

```

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

public void edit(Pessoa pessoa) {
    EntityManager em = null;
    try {
        em = getEntityManager();
        em.getTransaction().begin();
        pessoa = em.merge(pessoa);
        em.getTransaction().commit();
    } catch (Exception e) {
        throw new RuntimeException("Ocorreu um erro ao editar a pessoa.", e);
    } finally {
        if (em != null) {
            em.close();
        }
    }
}

public void destroy(Integer id) {
    EntityManager em = null;
    try {
        em = getEntityManager();
        em.getTransaction().begin();
        Pessoa pessoa;
        try {
            pessoa = em.getReference(Pessoa.class, id);
            pessoa.getIdPessoa();
        } catch (EntityNotFoundException enfe) {
            throw new RuntimeException("A pessoa com ID " + id + " não existe.", enfe);
        }
        em.remove(pessoa);
        em.getTransaction().commit();
    } finally {
        if (em != null) {
            em.close();
        }
    }
}
}

```

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 java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Query;
import javax.persistence.criteria.CriteriaQuery;
import model.Produto;

/**
 *
 */

```

```

* @author tonis
*/
public class ProdutoJpaController {

    private EntityManagerFactory emf;

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

    public List<Produto> findProdutoEntities() {
        EntityManager em = getEntityManager();
        try {
            CriteriaQuery<Produto> cq = em.getCriteriaBuilder().createQuery(Produto.class);
            cq.select(cq.from(Produto.class));
            Query q = em.createQuery(cq);
            return q.getResultList();
        } finally {
            em.close();
        }
    }

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

    public void edit(Produto produto) {
        EntityManager em = null;
        try {
            em = getEntityManager();
            em.getTransaction().begin();
            produto = em.merge(produto);
            em.getTransaction().commit();
        } catch (Exception ex) {
            if (em != null && em.getTransaction().isActive()) {
                em.getTransaction().rollback();
            }
            ex.printStackTrace();
        } finally {
            if (em != null) {
                em.close();
            }
        }
    }

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

```

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 javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.NoResultException;
import javax.persistence.Query;
import model.Usuario;

/**
 *
 * @author tonis
 */
public class UsuarioJpaController {

    private EntityManagerFactory emf;

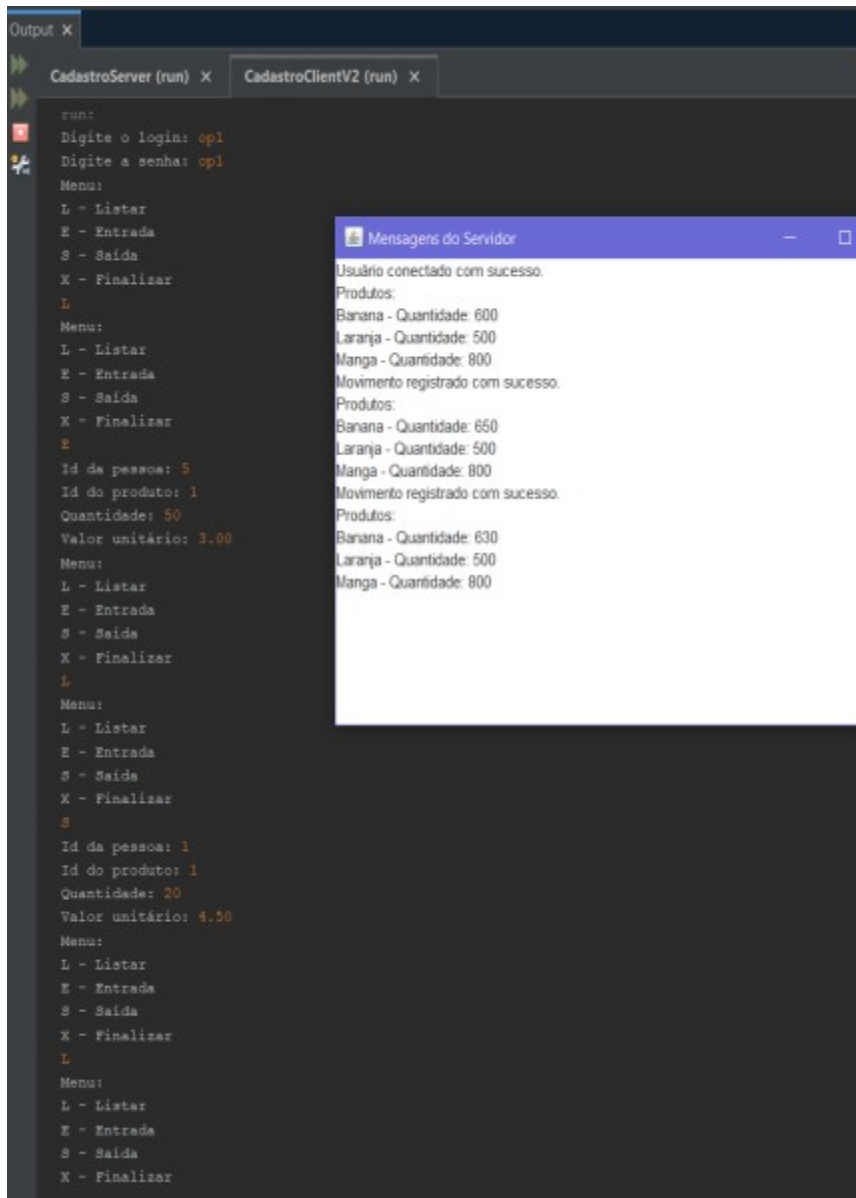
    public UsuarioJpaController(EntityManagerFactory emf) {
        this.emf = emf;
    }

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

    public Usuario findUsuario(String login, String senha) {
        EntityManager em = null;
        try {
            em = getEntityManager();
            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);
            return (Usuario) query.getSingleResult();
        } catch (NoResultException e) {
            return null;
        } finally {
            if (em != null) {
                em.close();
            }
        }
    }
}
```

* Resultados:

Teste de comandos:



```
run:
Digite o login: op1
Digite a senha: op1
Menu:
L - Listar
E - Entrada
S - Saída
X - Finalizar
L
Menu:
L - Listar
E - Entrada
S - Saída
X - Finalizar
E
Id da pessoa: 5
Id do produto: 1
Quantidade: 50
Valor unitário: 3.00
Menu:
L - Listar
E - Entrada
S - Saída
X - Finalizar
L
Menu:
L - Listar
E - Entrada
S - Saída
X - Finalizar
S
Id da pessoa: 1
Id do produto: 1
Quantidade: 20
Valor unitário: 4.50
Menu:
L - Listar
E - Entrada
S - Saída
X - Finalizar
L
Menu:
L - Listar
E - Entrada
S - Saída
X - Finalizar
```

Mensagens do Servidor

Usuário conectado com sucesso.

Produtos:

Banana - Quantidade: 600

Laranja - Quantidade: 500

Manga - Quantidade: 800

Movimento registrado com sucesso.

Produtos:

Banana - Quantidade: 650

Laranja - Quantidade: 500

Manga - Quantidade: 800

Movimento registrado com sucesso.

Produtos:

Banana - Quantidade: 630

Laranja - Quantidade: 500

Manga - Quantidade: 800

*** Análise e Conclusão:**

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

Threads permitem processar respostas do servidor de forma assíncrona, mantendo o thread principal livre para continuar outras tarefas enquanto espera a resposta.

Para isso, cada solicitação ao servidor precisa de uma nova thread, que cuida do envio, recebimento e processamento da resposta, sem interferir na execução do restante da aplicação.

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

O método `invokeLater` da classe `SwingUtilities` agenda tarefas para serem executadas no Event Dispatch Thread (EDT), que gerencia atualizações na interface gráfica do Swing.

Ele recebe um objeto `Runnable` com o código a ser executado no EDT, garantindo que alterações na GUI ocorram de forma segura, após o processamento de eventos pendentes.

c) Como os objetos são enviados e recebidos pelo Socket Java?

No Java, objetos são transmitidos por sockets através da serialização e desserialização. Serializar converte o objeto em bytes para envio pela rede, enquanto desserializar reconstrói o objeto a partir desses bytes recebidos.

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

Síncrono: Em sockets Java, operações síncronas fazem o cliente esperar pela resposta do servidor, bloqueando a thread até que a leitura ou escrita termine. Isso pode causar lentidão, já que a thread não pode executar outras tarefas enquanto espera.

Assíncrono: Com técnicas como threads separadas ou `java.nio`, o cliente pode continuar outras operações enquanto aguarda a resposta. O I/O não bloqueia a thread principal, tornando o sistema mais responsivo e eficiente para lidar com múltiplas conexões.