



Missão Prática - Nível 5 Mundo 3

Campus : POLO COPACABANA

Curso : DESENVOLVIMENTO FULL STACK

Disciplina : RPG0018 - Por que não paralelizar

Turma : 9003

Semestre: 2023.3 FLEX

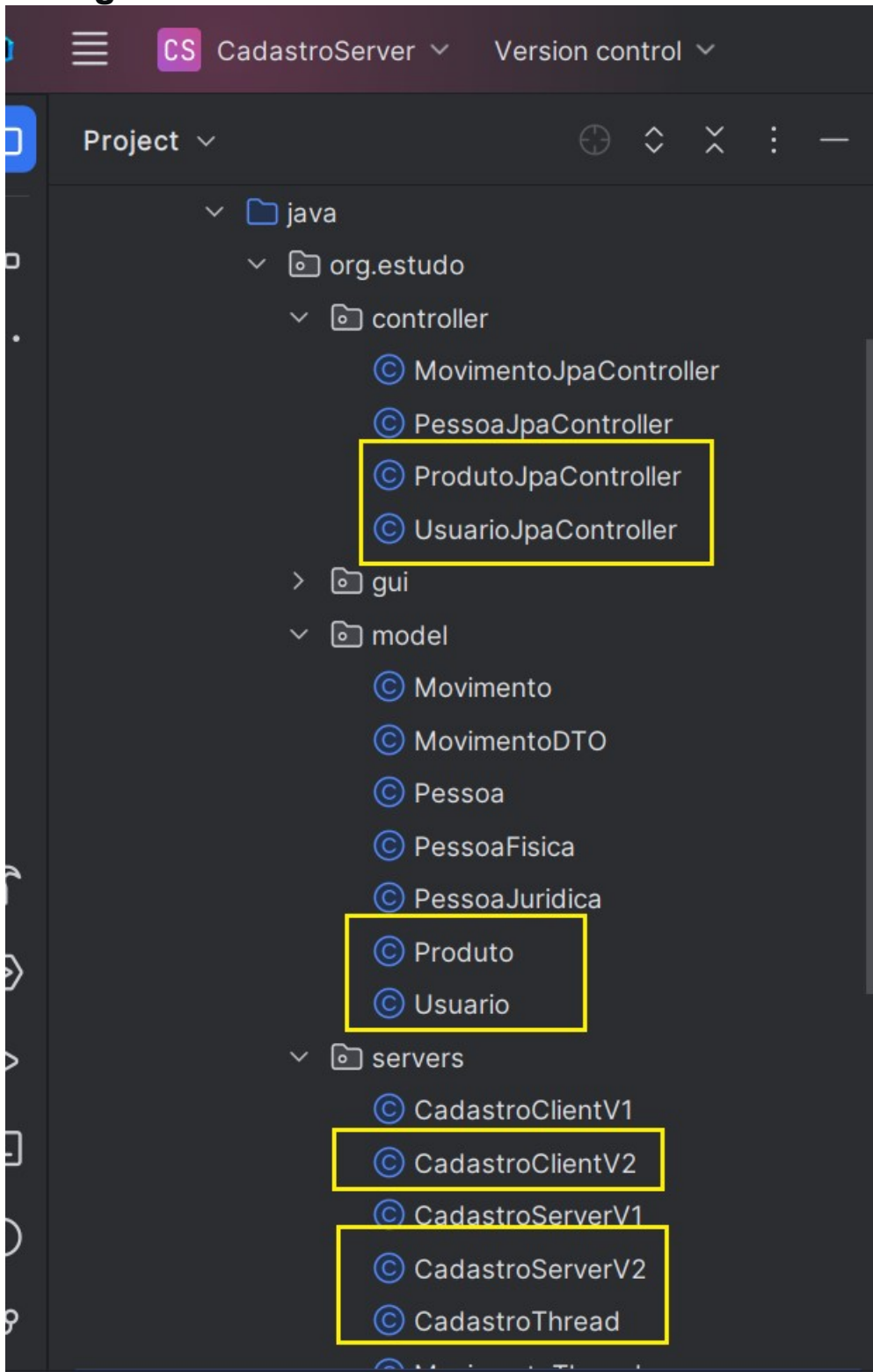
Nome : MARCO SERGIO ALBINO VITTORIO BAROZZI

1º Procedimento | Criando o Servidor e Cliente de Teste

Objetivos:

- 1 - Criar o projeto do servidor, utilizando o nome CadastroServer, do tipo console.
- 2 - Criar a camada de persistência em CadastroServer.
- 3 - Criar a camada de controle em CadastroServer
- 4 - No pacote principal, cadastroserver, adicionar a Thread de comunicação, com o nome CadastroThread.
- 5 - Implementar a classe de execução (main)
- 6 - Criar o cliente de teste, utilizando o nome CadastroClient, do tipo console, no modelo Ant padrão
- 7 - Configurar o projeto do cliente para uso das entidades
- 8 - Testar o sistema criado, com a execução dos dois projetos.

Códigos desenvolvidos :



Classe CadastroServerV1.java

```
package org.estudo.servers;

import org.estudo.controller.*;

import org.estudo.servers.MovimentoThread;

import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import java.io.IOException;
import java.io.InputStream;
import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.ServerSocket;
import java.net.Socket;

public class CadastroServerV2 {
    public static void main(String[] args) {
        // Verifica a presença do arquivo persistence.xml
        try {
            InputStream is =
Thread.currentThread().getContextClassLoader().getResourceAsStream("META-INF/persistence.xml");
            if (is == null) {
                System.out.println("Não foi possível encontrar o arquivo persistence.xml no classpath!");
            } else {
                System.out.println("Arquivo persistence.xml encontrado com sucesso no classpath.");
                is.close();
            }
        } catch (Exception e) {
            e.printStackTrace();
        }

        // Instancia um objeto do tipo EntityManagerFactory a partir da unidade de persistência.
        EntityManagerFactory emf =
Persistence.createEntityManagerFactory("jdbc/loja");

        MovimentoJpaController ctrlMov = new MovimentoJpaController(emf);
        PessoaJpaController ctrlPessoa = new PessoaJpaController(emf);

        System.out.println(" abaixo Declaração das variáveis JPAs");

        try {
            ServerSocket serverSocket = new ServerSocket(4321);
            System.out.println("ServerSocket iniciado na porta 4321");

            while (true) {
                //System.out.println("Conectado ao servidor
ANTES...serverSocket.accept();");
                Socket clientSocket = serverSocket.accept();
                //System.out.println("Conectado ao
servidor...serverSocket.accept();");
```

```

        ObjectOutputStream outServer = new
ObjectOutputStream(clientSocket.getOutputStream());
        ObjectInputStream inServer = new
ObjectInputStream(clientSocket.getInputStream());

        // Inicie diretamente a MovimentoThread
        Thread clientThread = new Thread(new MovimentoThread(ctrlMov,
ctrlPessoa, outServer, inServer));
        //System.out.println("Antes clientThread.start() MOVIMENTO");
        clientThread.start();
    }
} catch (IOException e) {
    e.printStackTrace();
}
}
}

```

Classe CadastroClientV1.java

```

package org.estudo.servers;

import org.estudo.model.Produto;

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

public class CadastroClientV1 {

    public static void main(String[] args) {
        try {
            // a. Instanciar um Socket apontando para localhost, na porta
4321.
            Socket socket = new Socket("localhost", 4321);

            // b. Encapsular os canais de entrada e saída do Socket
            ObjectOutputStream out = new
ObjectOutputStream(socket.getOutputStream());
            ObjectInputStream in = new
ObjectInputStream(socket.getInputStream());

            // c. Escrever o login e a senha na saída.
            out.writeObject("op1"); // Login
            out.writeObject("op1"); // Senha

            // d. Enviar o comando L no canal de saída.
            out.writeObject("L");

            // Receber a coleção de entidades no canal de entrada.
            System.out.println("Usuario conectado com sucesso");

            List<Produto> produtos = (List<Produto>) in.readObject();

            // e. Apresentar o nome de cada entidade recebida.
            for (Produto produto : produtos) {
                System.out.println(produto.getNome());
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```

    }

    // Fechar a conexão.
    socket.close();

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

}
}

```

Classe ProdutoJpaController.java

```

package org.estudo.controller;

import org.estudo.model.Produto;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.TypedQuery;

import java.util.List;

public class ProdutoJpaController {
    private EntityManagerFactory emf = null;

    public ProdutoJpaController(EntityManagerFactory emf) {
        this.emf = Persistence.createEntityManagerFactory("jdbc/loja");
    }

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

    public List<Produto> getAllProducts() {
        EntityManager em = getEntityManager();
        try {
            TypedQuery<Produto> query = em.createQuery("SELECT p FROM Produto
p", Produto.class);
            return query.getResultList();
        } finally {
            em.close();
        }
    }
}

```

Classe UsuarioJpaController.java

```

package org.estudo.controller;

import org.estudo.model.Usuario;
import javax.persistence.EntityManager;

```

```

import javax.persistence.EntityManagerFactory;
import javax.persistence.Persistence;
import javax.persistence.TypedQuery;

public class UsuarioJpaController {

    private EntityManagerFactory emf = null;

    public UsuarioJpaController(EntityManagerFactory emf) {
        this.emf = Persistence.createEntityManagerFactory("jdbc/loja");
    }

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

    public Usuario findUsuario(String login, String senha) {
        EntityManager em = getEntityManager();
        try {

            System.out.println("|||||==> Dentro de Usuario jpa controler
login "+login+", senha "+ senha);

            /*
            TypedQuery<Usuario> query = em.createQuery(
                "SELECT u FROM Usuario u WHERE u.login = :login AND
u.senha = :senha", Usuario.class);
            query.setParameter("login", login);
            query.setParameter("senha", senha);

            */
            TypedQuery<Usuario> query = em.createQuery(
                "SELECT u FROM Usuario u WHERE u.login = 'op1' AND
u.senha = 'op1'", Usuario.class);
            return query.getSingleResult();
        } catch (Exception e) {
            System.out.println("XXXX==> Dentro de Usuario jpa controler
ERRO ==> "+e.getMessage());
            return null;
        } finally {
            em.close();
        }
    }
}

```

Classe Produto.java

```

package org.estudo.model;

import javax.persistence.*;

import java.io.Serializable;

```

```

import java.math.BigDecimal;
import java.util.Objects;

@Entity
public class Produto implements Serializable {
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Id
    @Column(name = "idProduto")
    private int idProduto;
    @Basic
    @Column(name = "nome")
    private String nome;
    @Basic
    @Column(name = "quantidade")
    private Integer quantidade;
    @Basic
    @Column(name = "precoVenda")
    private BigDecimal precoVenda;

    public int getIdProduto() {
        return idProduto;
    }

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

    public String getNome() {
        return nome;
    }

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

    public Integer getQuantidade() {
        return quantidade;
    }

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

    public BigDecimal getPrecoVenda() {
        return precoVenda;
    }

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

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Produto produto = (Produto) o;
        return idProduto == produto.idProduto && Objects.equals(nome,

```

```

produto.nome) && Objects.equals(quantidade, produto.quantidade) &&
Objects.equals(precoVenda, produto.precoVenda);
    }

    @Override
    public int hashCode() {
        return Objects.hash(idProduto, nome, quantidade, precoVenda);
    }
}

```

Classe Usuario.Java

```

package org.estudo.model;

import javax.persistence.*;

import java.io.Serializable;
import java.util.Objects;

@Entity
public class Usuario implements Serializable {
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Id
    @Column(name = "idUsuario")
    private int idUsuario;
    @Basic
    @Column(name = "login")
    private String login;
    @Basic
    @Column(name = "senha")
    private String senha;

    public int getIdUsuario() {
        return idUsuario;
    }

    public void setIdUsuario(int 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;
    }

    @Override

```



```

    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Usuario usuario = (Usuario) o;
        return idUsuario == usuario.idUsuario && Objects.equals(login,
usuario.login) && Objects.equals(senha, usuario.senha);
    }

    @Override
    public int hashCode() {
        return Objects.hash(idUsuario, login, senha);
    }
}

```

Classe CadastroThread.java

```

package org.estudo.servers.thread;

import org.estudo.controller.ProdutoJpaController;
import org.estudo.controller.UsuarioJpaController;
import org.estudo.model.Usuario;

import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;

public class CadastroThread extends Thread {
    private ProdutoJpaController ctrl;
    private UsuarioJpaController ctrlUsu;
    private Socket s1;

    // Construtor
    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());

            // Obter o login e a senha
            String login = (String) in.readObject();
            String senha = (String) in.readObject();

            // Verificar o login
            Usuario user = ctrlUsu.findUsuario(login, senha);
            if (user == null) {
                out.writeObject("Usuário inválido.");
                s1.close();
                return;
            }
        }
    }
}

```

```

    }

    // Iniciar o loop de resposta
    while (true) {
        String command = (String) in.readObject();
        if ("L".equals(command)) {
            // Suponho que o método em ctrl retorne uma lista de
produtos
            out.writeObject(ctrl.getAllProducts());
        }
        // Você pode expandir isso para outros comandos conforme
necessário
    }
} catch (Exception e) {
    e.printStackTrace();
}
}
}

```

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="jdbc/loja" transaction-type="RESOURCE_LOCAL">
        <provider>org.eclipse.persistence.jpa.PersistenceProvider</provider>

        <class>org.estudo.model.Produto</class>

        <exclude-unlisted-classes>false</exclude-unlisted-classes>

        <properties>
            <property name="javax.persistence.jdbc.driver"
value="com.microsoft.sqlserver.jdbc.SQLServerDriver" />
            <property name="javax.persistence.jdbc.url"
value="jdbc:sqlserver://localhost:1434;databaseName=loja;encrypt=true;trustSe
rverCertificate=true" />
            <property name="javax.persistence.jdbc.user" value="loja" />
            <property name="javax.persistence.jdbc.password" value="loja" />

            <!-- EclipseLink settings -->
            <!-- EclipseLink log
            <property name="eclipselink.logging.level" value="FINE" />
            -->
            <property name="eclipselink.ddl-generation" value="none" />
        </properties>
    </persistence-unit>
</persistence>

```

Pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.1.4</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.estudo</groupId>
  <artifactId>CadastroServer</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>ServersThreads</name>
  <description>Estudo threads</description>
  <properties>
    <java.version>17</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>

    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
      <scope>runtime</scope>
      <optional>true</optional>
    </dependency>
    <dependency>
      <groupId>com.microsoft.sqlserver</groupId>
      <artifactId>mssql-jdbc</artifactId>
      <scope>runtime</scope>
    </dependency>
    <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
      <optional>true</optional>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>

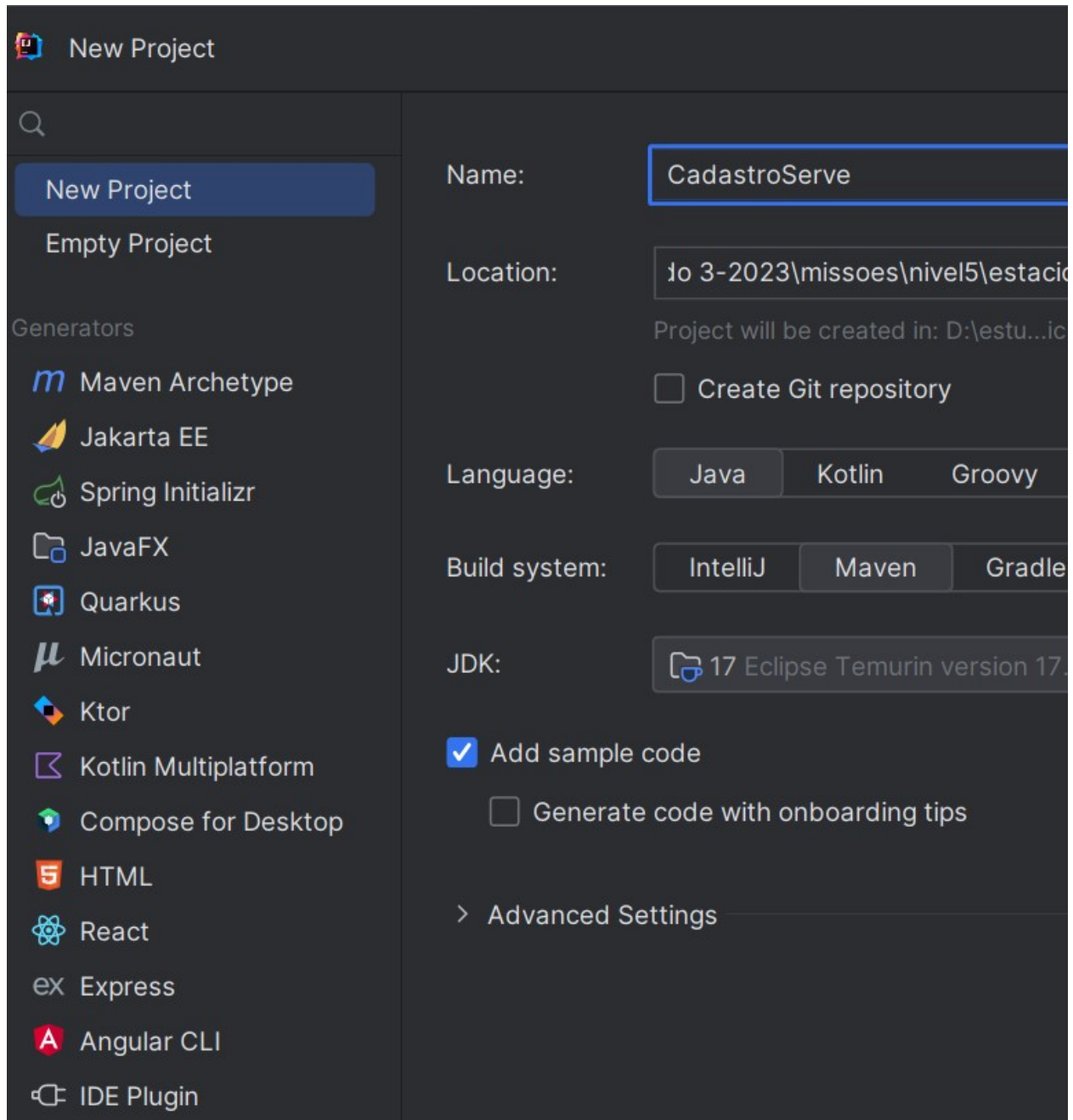
    <!-- Dependência do EclipseLink -->
    <dependency>
      <groupId>org.eclipse.persistence</groupId>
      <artifactId>eclipselink</artifactId>
      <version>2.7.9</version>
    </dependency>
  </dependencies>
</project>
```

```
<!-- Dependência da especificação JPA -->
<dependency>
  <groupId>javax.persistence</groupId>
  <artifactId>javax.persistence-api</artifactId>
  <version>2.2</version>
</dependency>
</dependencies>

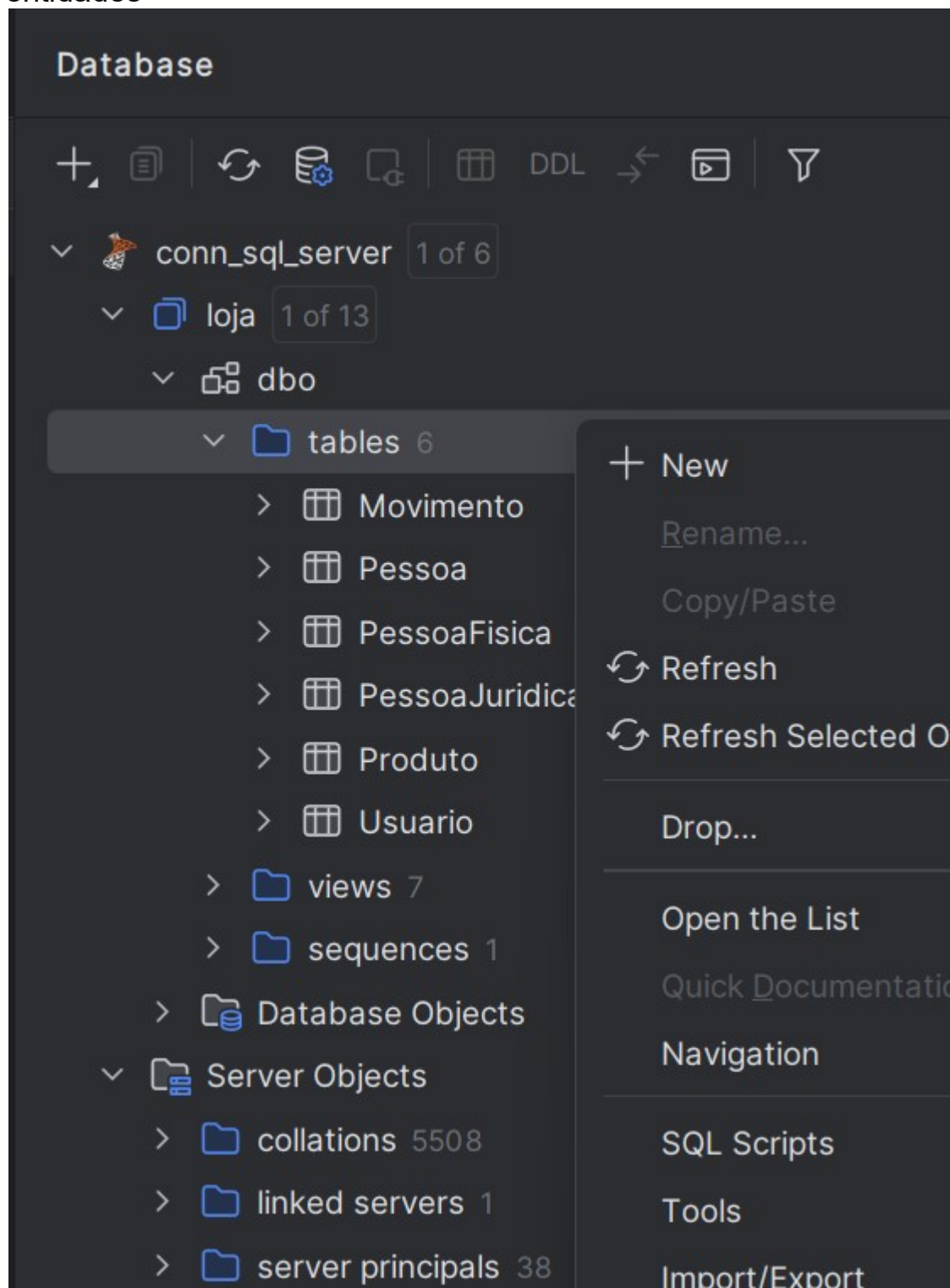
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
      <configuration>
        <excludes>
          <exclude>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
          </exclude>
        </excludes>
      </configuration>
    </plugin>
  </plugins>
</build>
</project>
```

Resultados da execução dos códigos

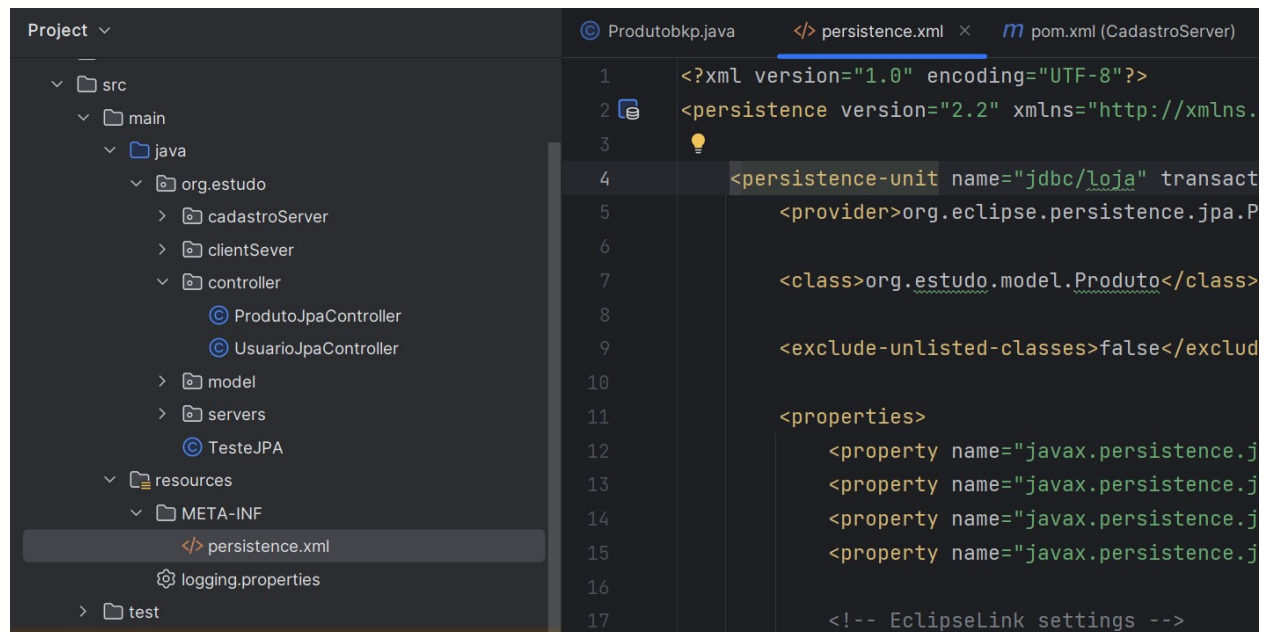
Criar o projeto



entidades



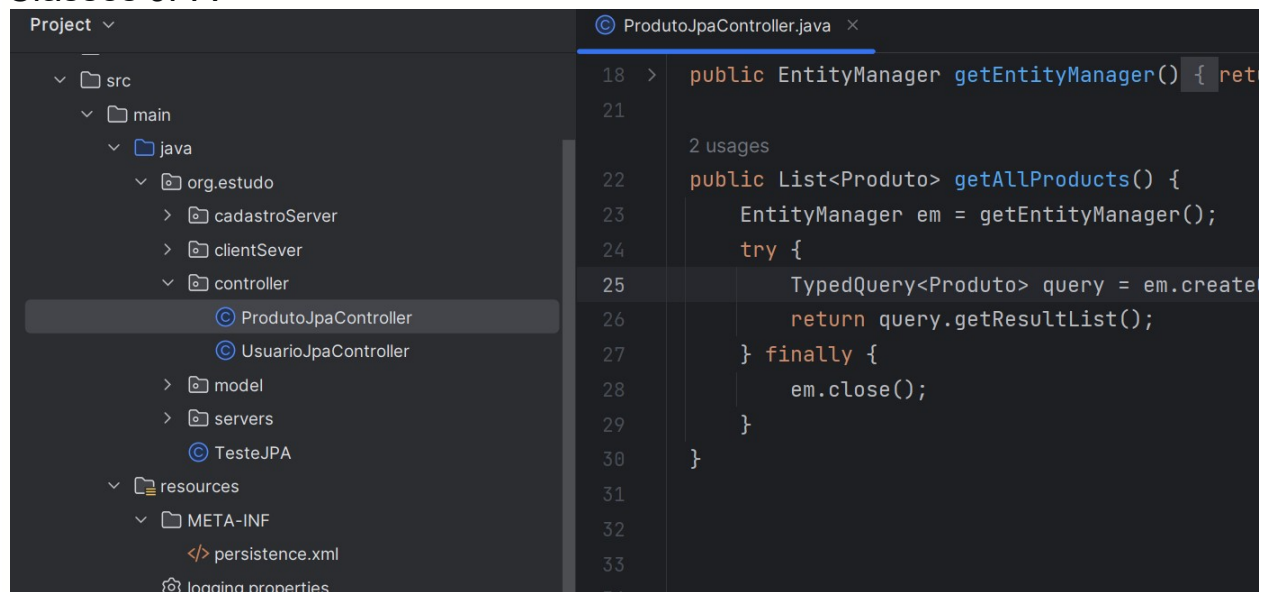
Dados conexão



The screenshot shows the Eclipse IDE with the 'Project' view on the left and the 'Code' editor on the right. The 'Project' view shows a project structure with a 'src' folder containing 'main' and 'resources' subfolders. The 'main' folder contains 'org.estudo' and 'resources' subfolders. The 'org.estudo' folder contains 'cadastroServer', 'clientSever', 'controller', 'model', and 'servers' subfolders. The 'controller' folder contains 'ProdutoJpaController' and 'UsuarioJpaController' files. The 'resources' folder contains 'META-INF' and 'logging.properties' files. The 'META-INF' folder contains 'persistence.xml' and 'logging.properties' files. The 'Code' editor shows the 'persistence.xml' file with the following content:

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <persistence version="2.2" xmlns="http://xmlns.ja
3
4 <persistence-unit name="jdbc/loja" transact
5
6 <provider>org.eclipse.persistence.jpa.P
7
8 <class>org.estudo.model.Produto</class>
9
10 <exclude-unlisted-classes>>false</exclud
11
12 <properties>
13 <property name="javax.persistence.j
14 <property name="javax.persistence.j
15 <property name="javax.persistence.j
16 <property name="javax.persistence.j
17 <!-- EclipseLink settings -->
```

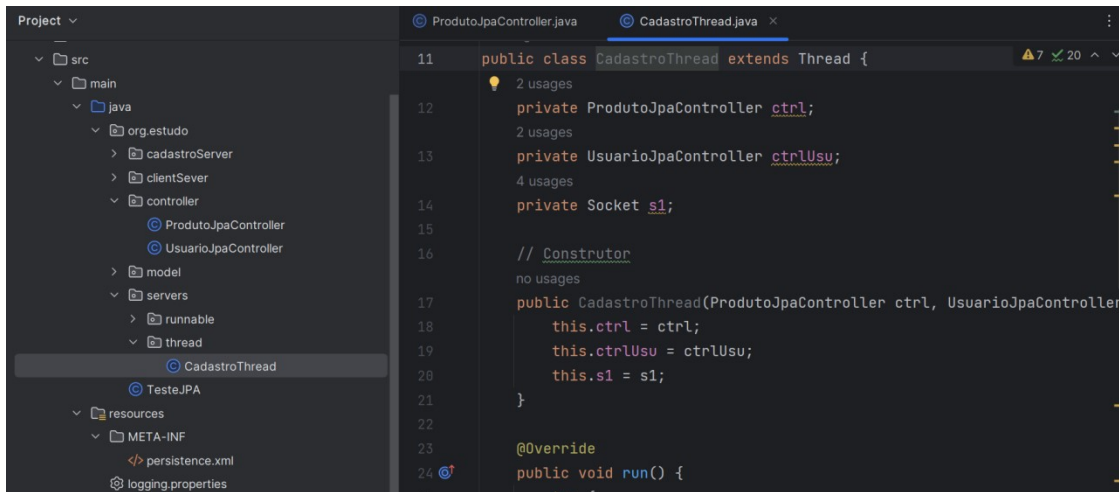
Classes JPA



The screenshot shows the Eclipse IDE with the 'Project' view on the left and the 'Code' editor on the right. The 'Project' view shows a project structure with a 'src' folder containing 'main' and 'resources' subfolders. The 'main' folder contains 'org.estudo' and 'resources' subfolders. The 'org.estudo' folder contains 'cadastroServer', 'clientSever', 'controller', 'model', and 'servers' subfolders. The 'controller' folder contains 'ProdutoJpaController' and 'UsuarioJpaController' files. The 'resources' folder contains 'META-INF' and 'logging.properties' files. The 'META-INF' folder contains 'persistence.xml' and 'logging.properties' files. The 'Code' editor shows the 'ProdutoJpaController.java' file with the following content:

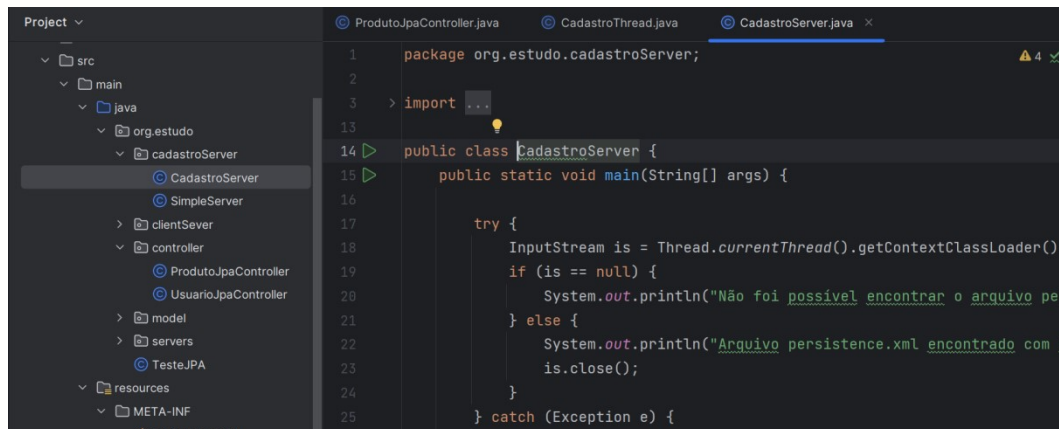
```
18 > public EntityManager getEntityManager() { ret
21
22 2 usages
23 public List<Produto> getAllProducts() {
24     EntityManager em = getEntityManager();
25     try {
26         TypedQuery<Produto> query = em.create
27         return query.getResultList();
28     } finally {
29         em.close();
30     }
31 }
32
33
```


Classe thread



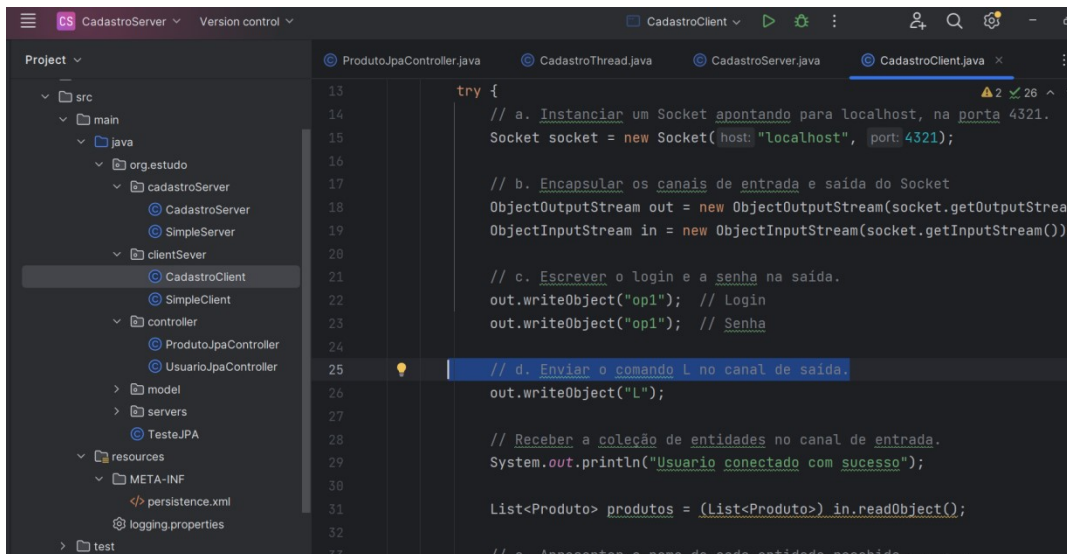
```
11 public class CadastroThread extends Thread {
12     private ProdutoJpaController ctrl;
13     private UsuarioJpaController ctrlUsu;
14     private Socket s1;
15
16     // Construtor
17     public CadastroThread(ProdutoJpaController ctrl, UsuarioJpaController ctrlUsu, Socket s1) {
18         this.ctrl = ctrl;
19         this.ctrlUsu = ctrlUsu;
20         this.s1 = s1;
21     }
22
23     @Override
24     public void run() {
```

Classes Main CadastroServer



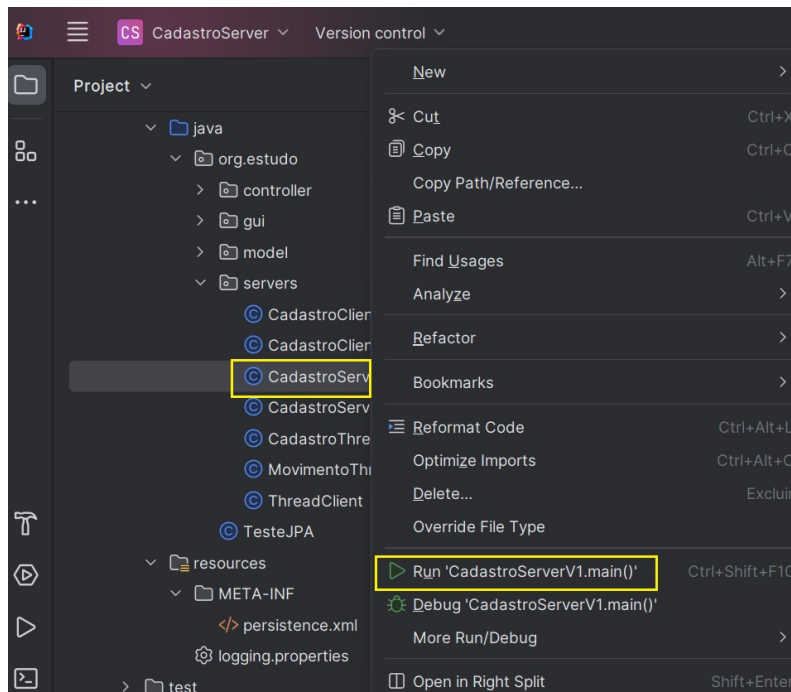
```
1 package org.estudo.cadastroServer;
2
3 import java.io.InputStream;
4
14 public class CadastroServer {
15     public static void main(String[] args) {
16
17         try {
18             InputStream is = Thread.currentThread().getContextClassLoader().getResourceAsStream("persistence.xml");
19             if (is == null) {
20                 System.out.println("Não foi possível encontrar o arquivo persistence.xml");
21             } else {
22                 System.out.println("Arquivo persistence.xml encontrado com sucesso");
23                 is.close();
24             }
25         } catch (Exception e) {
```

CadastroCliente

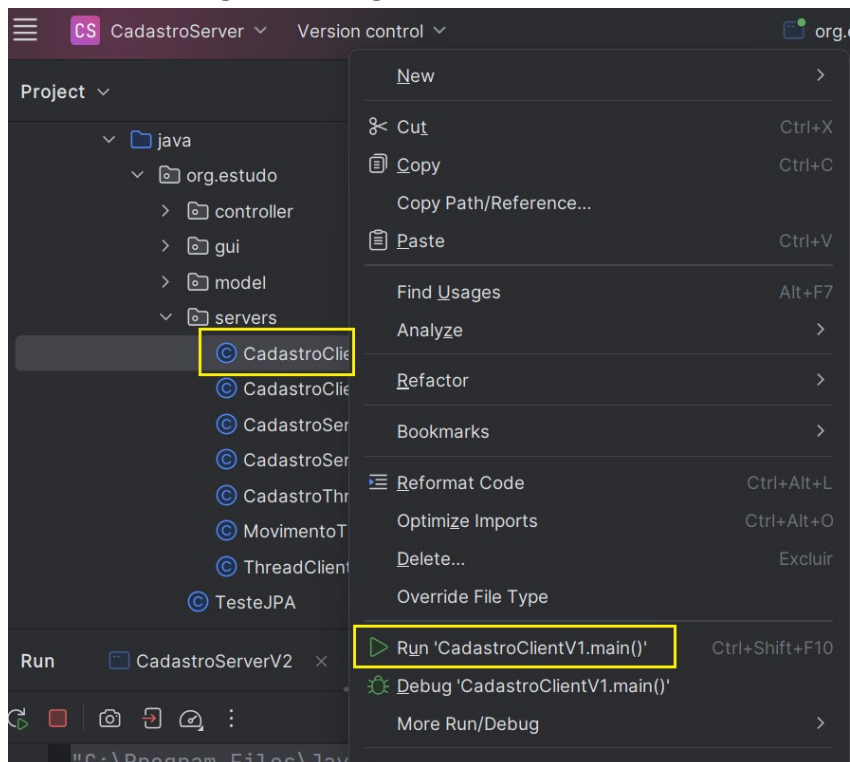


```
13 try {
14     // a. Instanciar um Socket apontando para localhost, na porta 4321.
15     Socket socket = new Socket("localhost", 4321);
16
17     // b. Encapsular os canais de entrada e saída do Socket
18     ObjectOutputStream out = new ObjectOutputStream(socket.getOutputStream());
19     ObjectInputStream in = new ObjectInputStream(socket.getInputStream());
20
21     // c. Escrever o login e a senha na saída.
22     out.writeObject("op1"); // Login
23     out.writeObject("op1"); // Senha
24
25     // d. Enviar o comando L no canal de saída.
26     out.writeObject("L");
27
28     // Receber a coleção de entidades no canal de entrada.
29     System.out.println("Usuario conectado com sucesso");
30
31     List<Produto> produtos = (List<Produto>) in.readObject();
32
33     // e. Apresentar o nome de cada entidade recebida.
```

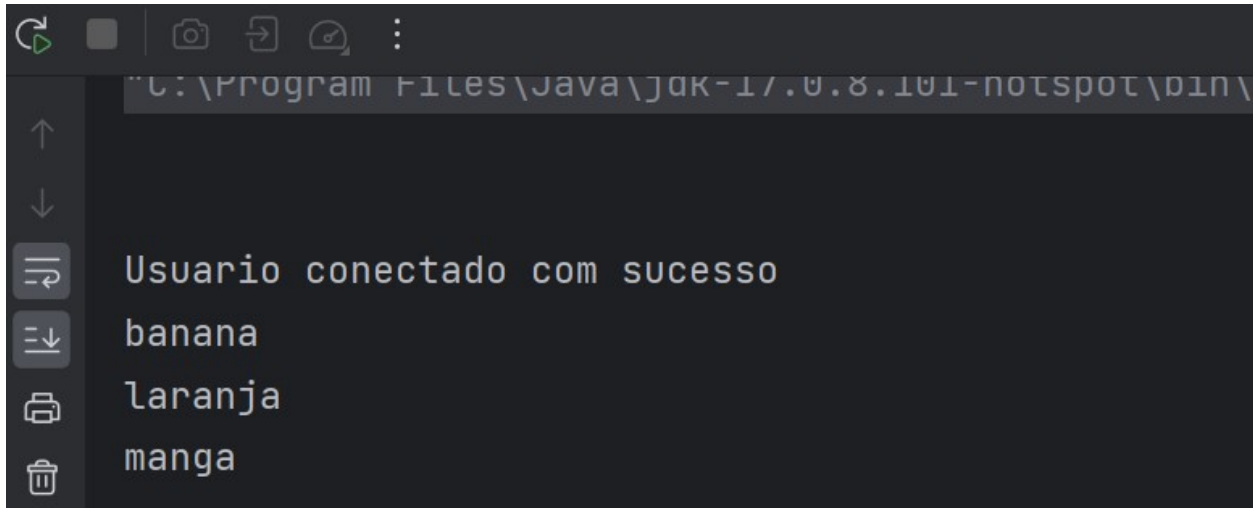

Executando CadastroServer



Executando CadastroClient



Resultado esperado

A screenshot of a Java IDE's console window. The title bar shows standard window controls and a file path: "C:\Program Files\Java\jdk-17.0.8.101-hotspot\bin\". The console output displays the following text: "Usuario conectado com sucesso", "banana", "laranja", and "manga". On the left side of the console, there is a vertical toolbar with icons for navigating through the output (up, down, first, last, previous, next) and a trash icon at the bottom.

Análise e Conclusão:

a) Como funcionam as classes Socket e ServerSocket?

ServerSocket é uma classe é usada para criar um servidor que espera por conexões de clientes. Essa classe cria um servidor que escuta em uma portNumbe por uma conexão de cliente onde a classe Socket representa essa conexão (seja do lado do cliente ou do servidor) Nessa classe Socket se faz a definição de um número da porta e nome do servidor será possível enviar e receber dados.

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

1- Multiplexação de serviços, pois permite muitos serviços ou aplicativos poder usar a rede simultaneamente.

2- Endereçamento de destino, pois identifica o serviço ou aplicativo específico nesse computador

3 -Gerenciamento de Sessão, durante uma comunicação, especialmente em protocolos como TCP, é importante manter o estado da sessão

4 –Segurança ou mecanismo de segurança.

5 - Convenções e Padrões, estas são reservadas para serviços específicos.

6 - Isolamento de problemas, se um serviço em particular falha ou tem um problema, ele não afeta outros serviços no mesmo sistema, pois cada serviço está vinculado a sua própria porta.

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

Servem para a serialização e deserialização de objetos em Java permitindo que se leia ou escreva objetos em fluxos, como arquivos ou conexões de rede.

A serialização é o processo de converter o estado de um objeto em uma sequência de bytes para armazenar ou transmitir para a memória, um banco de dados, ou um arquivo.

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

A JPA no cliente não garante, por si só, o isolamento do acesso ao banco de dados. O isolamento verdadeiro é geralmente conseguido por uma combinação de várias práticas e arquiteturas como arquitetura em camadas, serviços web ou APIs, segurança no banco de dados e isolamento de transações.