



Desenvolvimento Full Stack

Felipe Marques de Almeida - 202208291929

Polo Rodovia Br 407 - Juazeiro - Ba

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

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

Objetivo da Prática

Criar servidores Java com base em Sockets.

Criar clientes síncronos para servidores com base em Sockets.

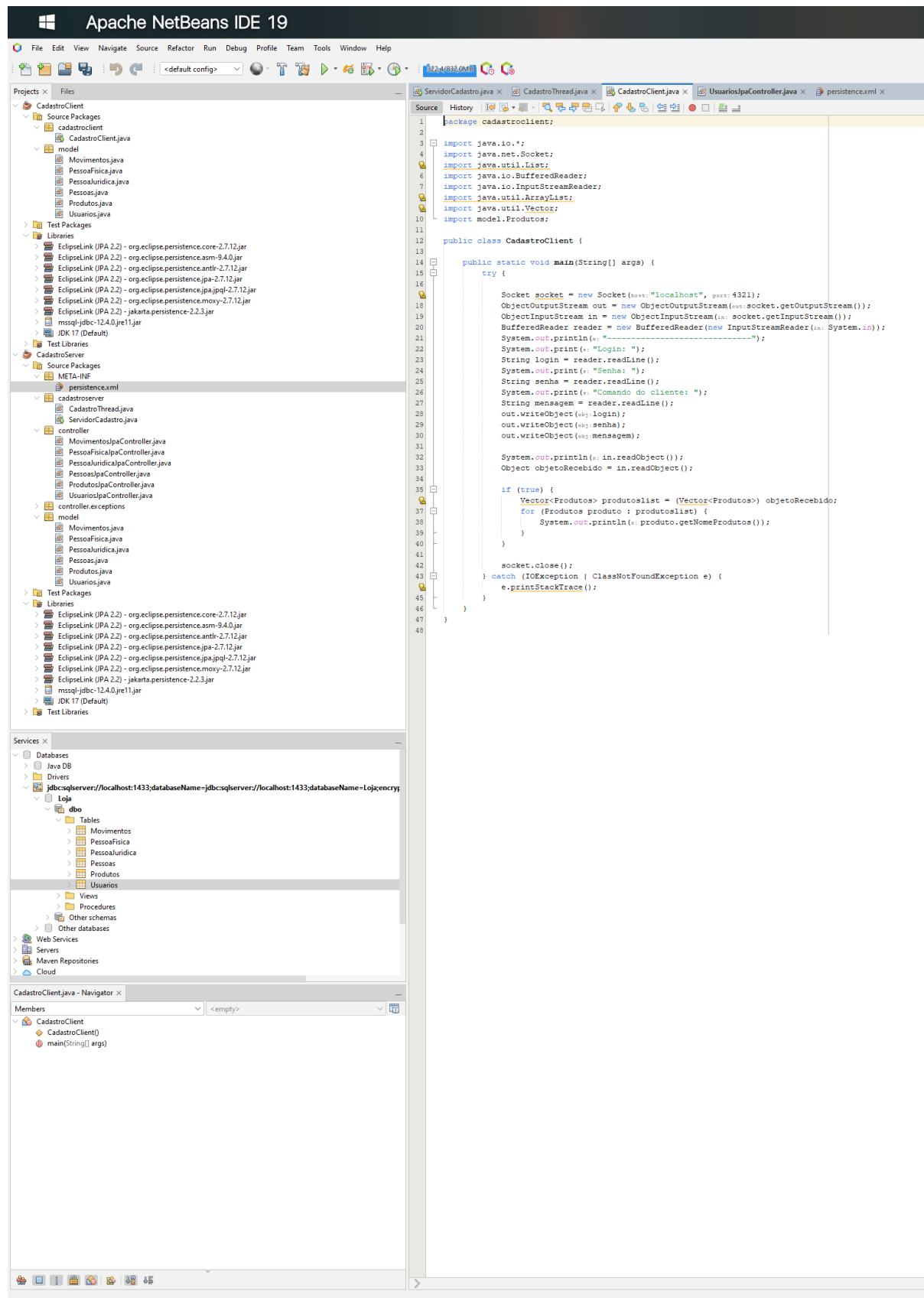
Criar clientes assíncronos para servidores com base em Sockets.

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.

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

CadastroClient:



CadastroThread:

Apache NetBeans IDE 19

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

545.6/832.0MB

Projects Files

CadastroClient

Source Packages

cadastroclient

CadastroClient.java

model

Movimentos.java

PessoaFisica.java

PessoaJuridica.java

Pessoas.java

Produtos.java

Usuarios.java

Test Packages

EclipseLink (IPA 2.2) - org.eclipse.persistence.core-2.7.12.jar

EclipseLink (IPA 2.2) - org.eclipse.persistence.asm-9.4.0.jar

EclipseLink (IPA 2.2) - org.eclipse.persistence.antlr-2.7.12.jar

EclipseLink (IPA 2.2) - org.eclipse.persistence.jpa-2.7.12.jar

EclipseLink (IPA 2.2) - org.eclipse.persistence.jpa.jpql-2.7.12.jar

EclipseLink (IPA 2.2) - org.eclipse.persistence.moxy-2.7.12.jar

EclipseLink (IPA 2.2) - jakarta.persistence-2.2.3.jar

mysql-jdbc-12.4.0.jre11.jar

JDK 17 (Default)

Test Libraries

CadastroServer

Source Packages

META-INF

persistence.xml

cadastroserver

CadastroThread.java

ServidorCadastro.java

controller

MovimentosJpaController.java

PessoaFisicaJpaController.java

PessoaJuridicaJpaController.java

PessoasJpaController.java

ProdutosJpaController.java

UsuariosJpaController.java

controller.exceptions

model

Movimentos.java

PessoaFisica.java

PessoaJuridica.java

Pessoas.java

Produtos.java

Usuarios.java

Test Packages

EclipseLink (IPA 2.2) - org.eclipse.persistence.core-2.7.12.jar

EclipseLink (IPA 2.2) - org.eclipse.persistence.asm-9.4.0.jar

EclipseLink (IPA 2.2) - org.eclipse.persistence.antlr-2.7.12.jar

EclipseLink (IPA 2.2) - org.eclipse.persistence.jpa-2.7.12.jar

EclipseLink (IPA 2.2) - org.eclipse.persistence.jpa.jpql-2.7.12.jar

EclipseLink (IPA 2.2) - org.eclipse.persistence.moxy-2.7.12.jar

EclipseLink (IPA 2.2) - jakarta.persistence-2.2.3.jar

mysql-jdbc-12.4.0.jre11.jar

JDK 17 (Default)

Test Libraries

Services

Databases

Java DB

Drivers

jdbc:sqlserver://localhost:1433;databaseName=jdbcsqlserver//localhost:1433;databaseName=Lojencry

Loja

dbo

Tables

Movimentos

PessoaFisica

PessoaJuridica

Pessoas

Produtos

Usuarios

Views

Procedures

Other schemas

Other databases

Web Services

Servers

Maven Repositories

Cloud

Navigator

Members

CadastroThread: Thread

CadastroThread(ProdutosJpaController ctrl, UsuariosJpaController ctrlUsu, Socket socket)

isValidUser(String login, String senha): boolean

run(): Thread

ctrl: ProdutosJpaController

ctrlUsu: UsuariosJpaController

s1: Socket

Source

History

package cadastroserver;

import java.io.*;

import java.net.Socket;

import controller.UsuariosJpaController;

import controller.ProdutosJpaController;

public class CadastroThread extends Thread {

private final ProdutosJpaController ctrl;

private final UsuariosJpaController ctrlUsu;

private final Socket s1;

public CadastroThread(ProdutosJpaController ctrl, UsuariosJpaController ctrlUsu, Socket socket) {

this.ctrl = ctrl;

this.ctrlUsu = ctrlUsu;

this.s1 = socket;

}

@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();

if (isValidUser(login, senha)) {

out.writeObject(s1, "Login Correto!");

} else {

try (s1) {

out.writeObject(s1, "Credenciais inválidas. Encerrando conexão...");

}

String command;

while ((command = (String) in.readObject()) != null) {

if (command.equals(s1, "L")) {

out.writeObject(s1, ctrl.getProdutos());

}

} catch (IOException | ClassNotFoundException e) {

}

}

private boolean isValidUser(String login, String senha) {

return ctrlUsu.findUsuario(login, senha) != null;

}

}

ServidorCadastro:

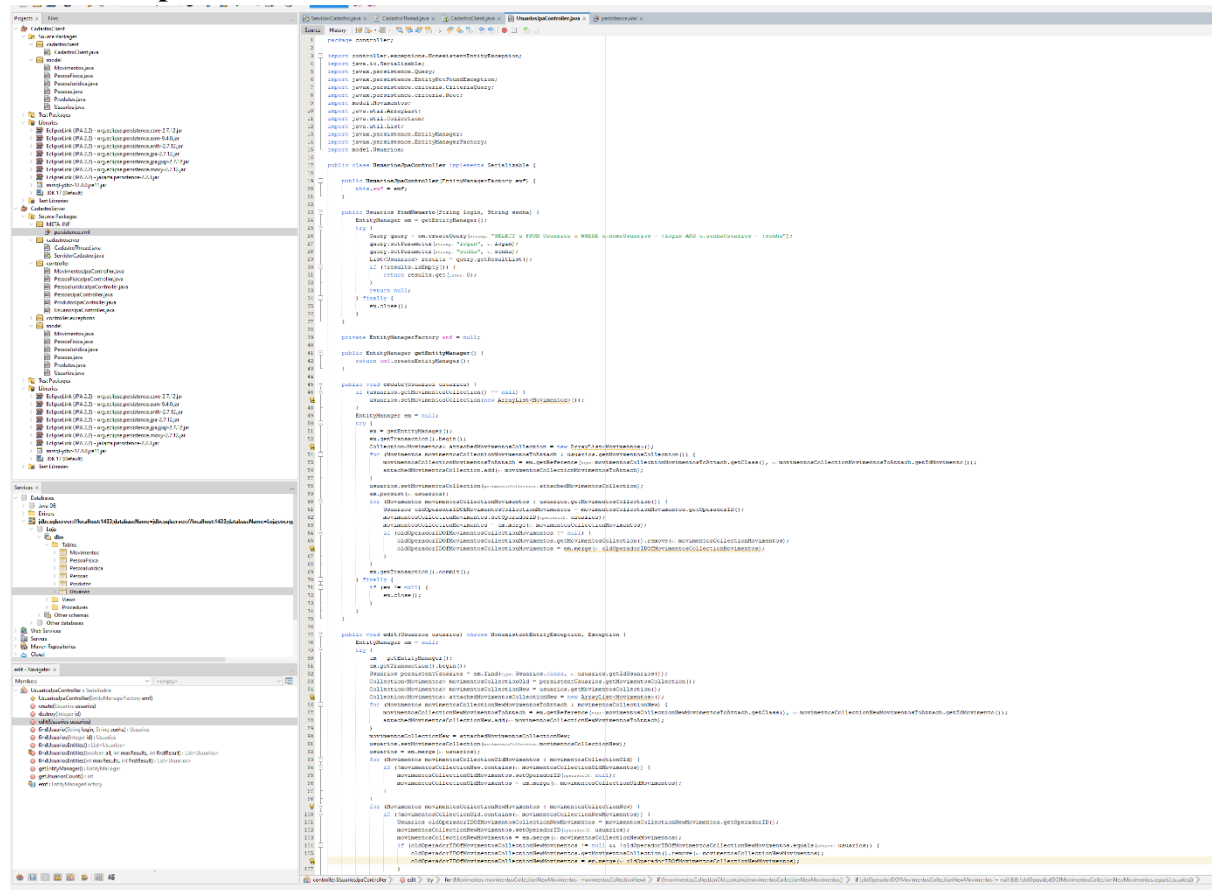
The screenshot displays the Apache NetBeans IDE 19 interface. The main window is divided into several panes:

- Project Explorer (Left):** Shows the project structure for 'CadastroServer'. It includes source packages like 'cadastroserver', 'controller', and 'model', along with test packages and libraries. The 'persistence.xml' file is highlighted under the 'META-INF' directory.
- Services (Left):** A window showing database connections. It lists a 'jdbc:sqlserver://localhost:1433;databaseName=jdbcsqlserver;localhost:1433;databaseName=Lojaencyr' connection, which is expanded to show tables like 'Movimentos', 'PessoaFisica', 'PessoaJuridica', 'Pessoas', 'Produtos', and 'Usuarios'.
- Source Editor (Right):** Displays the code for 'ServidorCadastro.java'. The code is as follows:

```
1 package cadastroserver;
2
3 import java.io.IOException;
4 import java.net.ServerSocket;
5 import java.net.Socket;
6 import javax.persistence.EntityManagerFactory;
7 import javax.persistence.Persistence;
8 import controller.ProdutosPaController;
9 import controller.UsuariosPaController;
10
11 public class ServidorCadastro {
12
13     public static void main(String[] args) {
14
15         EntityManagerFactory emf = Persistence.createEntityManagerFactory("CadastroServerPU");
16
17         ProdutosPaController ctrlProdutos = new ProdutosPaController(emf);
18         UsuariosPaController ctrlUsuarios = new UsuariosPaController(emf);
19
20         ServerSocket serverSocket = null;
21         try {
22             serverSocket = new ServerSocket(4321);
23             System.out.println("Servidor aguardando conexões na porta 4321...");
24         } catch (IOException e) {
25             e.printStackTrace();
26             System.exit(1);
27         }
28
29         while (true) {
30             try {
31                 Socket socket = serverSocket.accept();
32
33                 CadastroThread cadastroThread = new CadastroThread(ctrlProdutos, ctrlUsuarios, socket);
34                 cadastroThread.start();
35             } catch (IOException e) {
36                 e.printStackTrace();
37             }
38         }
39     }
40 }
41
42 }
```

The bottom status bar shows the file 'ServidorCadastro.java' is open, with line 45 highlighted.

UsuariosJpaController:



Apache NetBeans IDE 19

File Edit View Navigate Source Refactor Run Debug Profile Team Tools Window Help

Server: CadastroServer - App

Projects: Files

- CadastroClient
 - Source Packages
 - CadastroClient
 - model
 - Movimentos.java
 - PessoaFisica.java
 - PessoaJuridica.java
 - Pessoa.java
 - Produtos.java
 - Usuarios.java
 - Test Packages
 - Libraries
 - EclipseLink (PA 2.2) - org.eclipse.persistence.core-2.7.12.jar
 - EclipseLink (PA 2.2) - org.eclipse.persistence.asm-9.4.0.jar
 - EclipseLink (PA 2.2) - org.eclipse.persistence.antlr-2.7.12.jar
 - EclipseLink (PA 2.2) - org.eclipse.persistence.jpa-2.7.12.jar
 - EclipseLink (PA 2.2) - org.eclipse.persistence.jpa.pq-2.7.12.jar
 - EclipseLink (PA 2.2) - org.eclipse.persistence.moxy-2.7.12.jar
 - EclipseLink (PA 2.2) - jakarta.persistence-2.2.3.jar
 - msuaj-jdbc-12.4.0-jet11.jar
 - JDK 17 (Default)
- CadastroServer
 - Source Packages
 - persistence.xml
 - cadastroserver
 - CadastroThread.java
 - Server/Cadastro.java
 - controller
 - MovimentosController.java
 - PessoaFisicaController.java
 - PessoaJuridicaController.java
 - PessoaController.java
 - ProdutosController.java
 - UsuariosController.java
 - controller.exceptions
 - model
 - Movimentos.java
 - PessoaFisica.java
 - PessoaJuridica.java
 - Pessoa.java
 - Produtos.java
 - Usuarios.java
 - Test Packages
 - Libraries
 - EclipseLink (PA 2.2) - org.eclipse.persistence.core-2.7.12.jar
 - EclipseLink (PA 2.2) - org.eclipse.persistence.asm-9.4.0.jar
 - EclipseLink (PA 2.2) - org.eclipse.persistence.antlr-2.7.12.jar
 - EclipseLink (PA 2.2) - org.eclipse.persistence.jpa-2.7.12.jar
 - EclipseLink (PA 2.2) - org.eclipse.persistence.jpa.pq-2.7.12.jar
 - EclipseLink (PA 2.2) - org.eclipse.persistence.moxy-2.7.12.jar
 - EclipseLink (PA 2.2) - jakarta.persistence-2.2.3.jar
 - msuaj-jdbc-12.4.0-jet11.jar
 - JDK 17 (Default)

Services:

- Databases
 - Java DB
 - Driver
 - jdbc:server=localhost:1433;databaseName=jdbc:server=localhost:1433;databaseName=Loja-mery
 - Loja
 - Tables
 - Movimentos
 - PessoaFisica
 - PessoaJuridica
 - Pessoas
 - Produtos
 - Usuarios
 - Views
 - Procedures
 - Other schemes
 - Other databases
 - Web Services
 - Servers
 - Maps Repositories
 - Cloud

edit - Navigator

Members: Serializable empty

- UsuariosController (EntityManagerFactory em)
 - create(Usuarios usuarios)
 - destroy(Integer id)
 - edit(Usuarios usuarios)
 - findUsuario(String login, String senha) Usuarios
 - findUsuarios(Integer id) Usuarios
 - findUsuariosEnties() List<Usuarios>
 - findUsuariosEnties(boolean all, int maxResults, int firstResult) List<Usuarios>
 - findUsuariosEnties(int maxResults, int firstResult) List<Usuarios>
 - getEntityManager() EntityManager
 - getUsuariosCount() int
 - emf: EntityManagerFactory

Source

```
oldOperatorIDOfMovimentosCollectionNewMovimentos = em.merge(oldOperatorIDOfMovimentosCollectionNewMovimentos);
}
}
em.getTransaction().commit();
} catch (Exception ex) {
String msg = ex.getMessage();
if (msg != null && msg.length() > 0) {
Integer id = usuarios.getId();
if (findUsuarios(id) == null) {
throw new NonexistentEntityException("The usuarios with id " + id + " no longer exists.");
}
}
throw ex;
} finally {
if (em != null) {
em.close();
}
}
}

public void destroy(Integer id) throws NonexistentEntityException {
EntityManager em = null;
try {
em = getEntityManager();
em.getTransaction().begin();
Usuarios usuarios;
try {
usuarios = em.getReference(Usuarios.class, id);
usuarios.getId();
} catch (EntityNotFoundException enfe) {
throw new NonexistentEntityException("The usuarios with id " + id + " no longer exists.", enfe);
}
Collection<Movimentos> movimentosCollection = usuarios.getMovimentosCollection();
for (Movimentos movimentosCollectionMovimentos : movimentosCollection) {
movimentosCollectionMovimentos.setOperatorID(null);
movimentosCollectionMovimentos = em.merge(movimentosCollectionMovimentos);
}
em.remove(usuarios);
em.getTransaction().commit();
} finally {
if (em != null) {
em.close();
}
}
}

public List<Usuarios> findUsuariosEnties() {
return findUsuariosEnties(true, null, -1, firstResult:-1);
}

public List<Usuarios> findUsuariosEnties(int maxResults, int firstResult) {
return findUsuariosEnties(false, maxResults, firstResult);
}

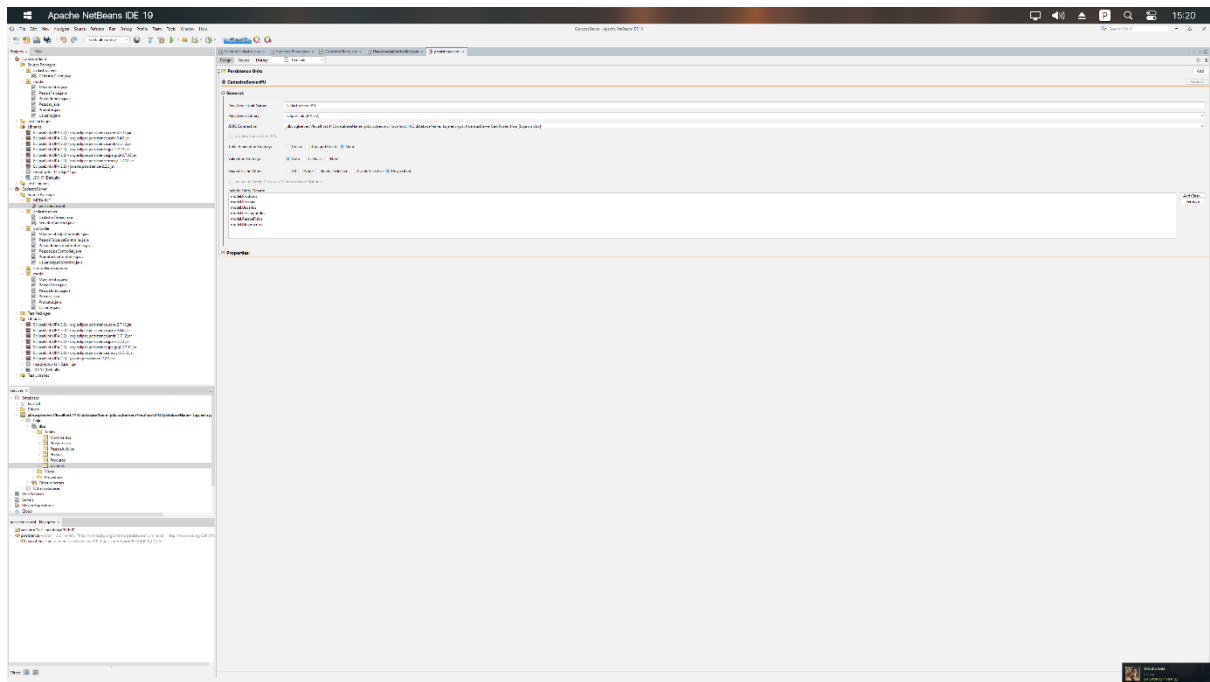
private List<Usuarios> findUsuariosEnties(boolean all, int maxResults, int firstResult) {
try {
CriteriaQuery cq = em.getCriteriaBuilder().createQuery();
cq.select(cq.from(Usuarios.class));
Query q = em.createQuery(cq);
if (all) {
q.setFirstResult(0);
} else {
q.setFirstResult(firstResult);
}
return q.getResultList();
} finally {
em.close();
}
}

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

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

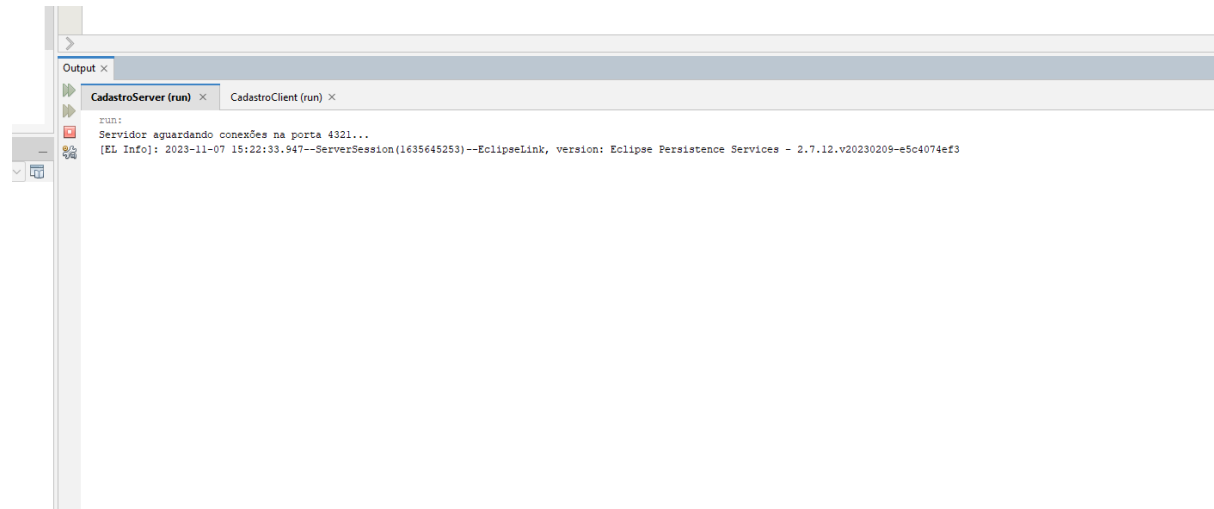
controller: UsuariosController edit try for (Movimentos movimentosCollectionNewMovimentos : movimentosCollectionNew) { if (movimentosCollectionOld.contains(movimentosCollectionNewMovimentos))

Persistence.xml:

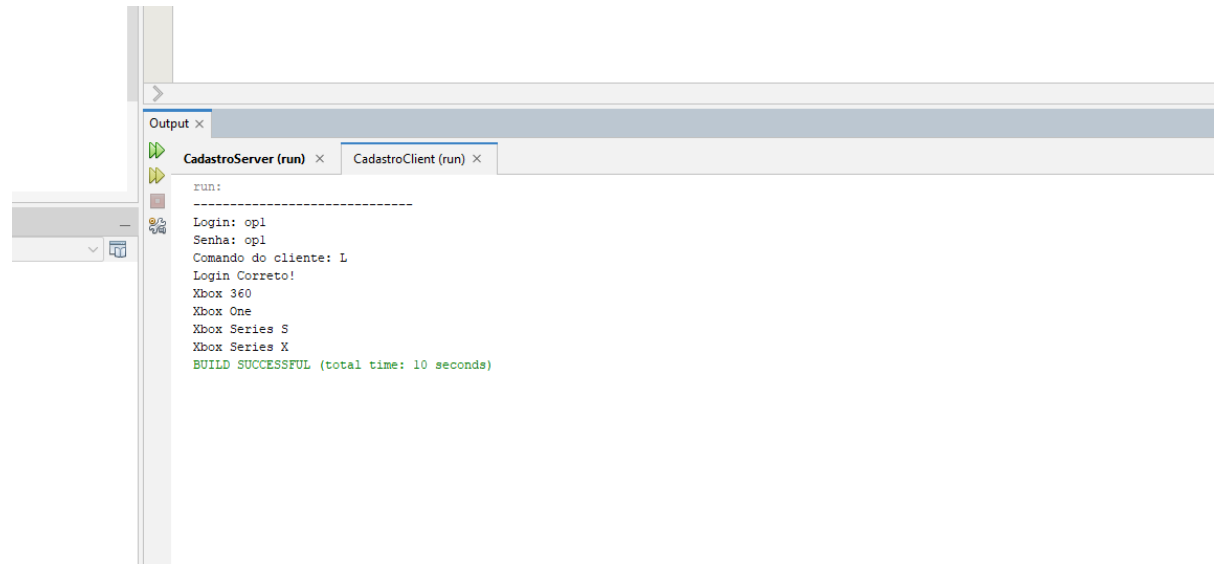


Resultados:

RunServidor:



RunClient:



Análise e Conclusão:

As classes `Socket` e `ServerSocket` desempenham papéis fundamentais na comunicação entre sistemas em uma rede. O `Socket` é usado para criar um ponto de comunicação em um determinado dispositivo, enquanto o `ServerSocket` age como o "ouvinte" que aguarda e aceita solicitações de conexão. Quando um cliente deseja se comunicar com um servidor, ele inicia uma conexão com o `ServerSocket`, que permite a transferência de dados entre as partes.

As portas desempenham um papel crucial na conexão com servidores, pois elas fornecem um meio de direcionar o tráfego para os aplicativos corretos em um servidor. Cada aplicativo que aceita conexões de rede geralmente usa uma porta específica para receber e processar dados. Portanto, as portas garantem que os dados sejam direcionados ao aplicativo correto no servidor.

As classes de entrada e saída `ObjectInputStream` e `ObjectOutputStream` são usadas para transmitir objetos entre sistemas. Elas são importantes porque permitem a comunicação de dados complexos em formato de objeto. Além disso, a serialização é necessária para que os objetos sejam transmitidos de forma eficaz e reconstruídos com precisão no destino. A serialização converte objetos em uma sequência de bytes que podem ser transmitidos pela rede e posteriormente desserializados para recriar o objeto original.

Ao utilizar as classes de entidades JPA no cliente, é possível garantir o isolamento do acesso ao banco de dados devido ao conceito de persistência de dados. O JPA (Java Persistence API) fornece uma camada de abstração sobre o banco de dados, permitindo que as operações de banco de dados sejam realizadas sem a necessidade de escrever SQL diretamente. Isso ajuda a manter o acesso aos dados consistente e protegido, evitando que o cliente acesse diretamente o banco de dados subjacente. Portanto, mesmo no cliente, o acesso aos dados é gerenciado de forma isolada e controlada, seguindo os princípios de persistência e abstração de banco de dados.