Wellington Cesar Fonseca

DESENVOLVIMENTO FULL STACK

MISSÃO PRÁTICA | NÍVEL 3 | MUNDO 5

RPG0018 - POR QUE NÃO PARALELIZAR

OBJETIVO

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.

2º PROCEDIMENTO | SERVIDOR COMPLETO E CLIENTE ASSÍNCRONO

Como as Threads podem ser utilizadas para o tratamento assíncrono das respostas enviadas pelo servidor?
R:
Threads permitem que o servidor processe múltiplas solicitações simultaneamente sem bloquear o fluxo principal de execução. Isso é crucial para operações assíncronas, permitindo que o servidor continue a responder a novas requisições enquanto processa respostas em paralelo.

Para que serve o método invokeLater, da classe SwingUtilities?
R:
É usado para colocar um bloco de código na fila de eventos do Swing para execução posterior, garantindo que as atualizações da interface gráfica (GUI) ocorram na Thread de Despacho de Eventos, evitando problemas de concorrência e mantendo a GUI responsiva.

Como os objetos são enviados e recebidos pelo Socket Java?
R:
Objetos são enviados e recebidos por meio de ObjectOutputStream e ObjectInputStream, respectivamente. Eles permitem que objetos serializáveis sejam convertidos em um fluxo de bytes para transmissão através de Sockets e reconstruídos no lado do receptor.

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

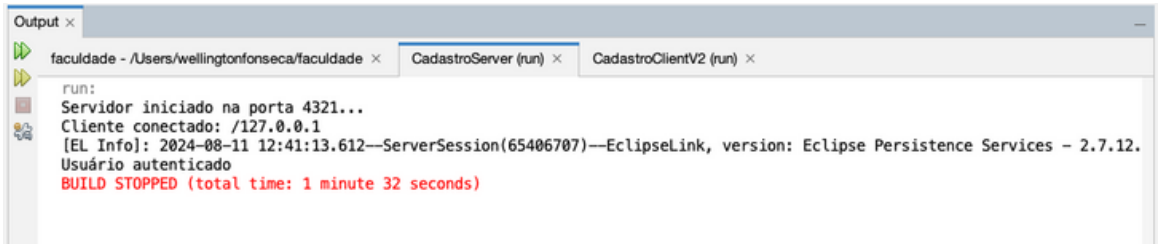
- **Síncrono:** O cliente aguarda a resposta do servidor antes de continuar o processamento. Isso pode causar bloqueios e reduzir a eficiência se o servidor demorar a responder.
- **Assíncrono:** Permite que o cliente continue o processamento sem esperar pela resposta do servidor. Isso melhora a responsividade, pois o cliente pode executar outras tarefas enquanto aguarda a conclusão da operação de rede, mas requer gerenciamento de estado e lógica adicional para lidar com as respostas.

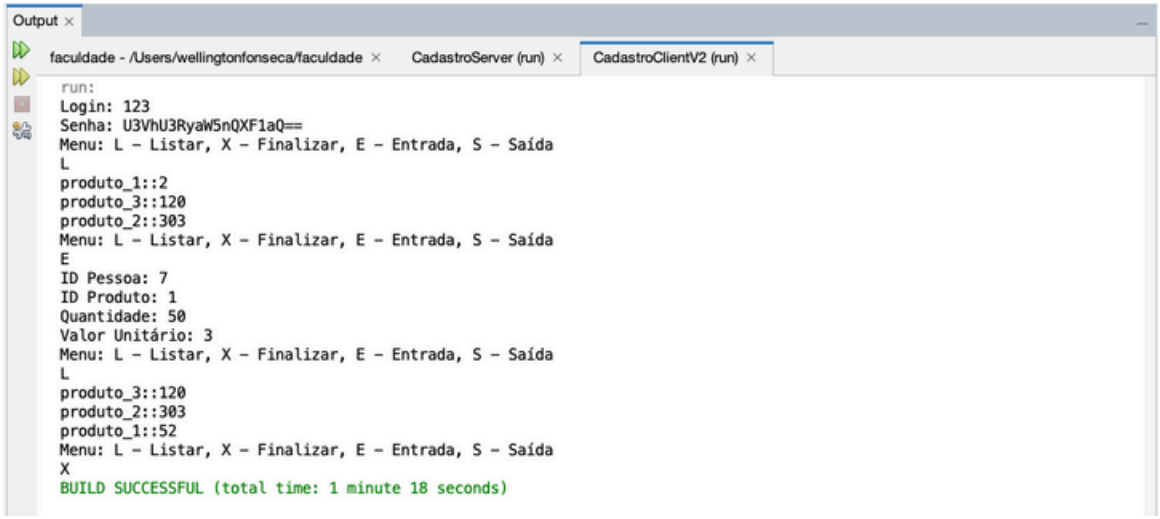
IMPORTANTE!
Tive que remover parte do persistence, pois o github está rejeitando o push por:

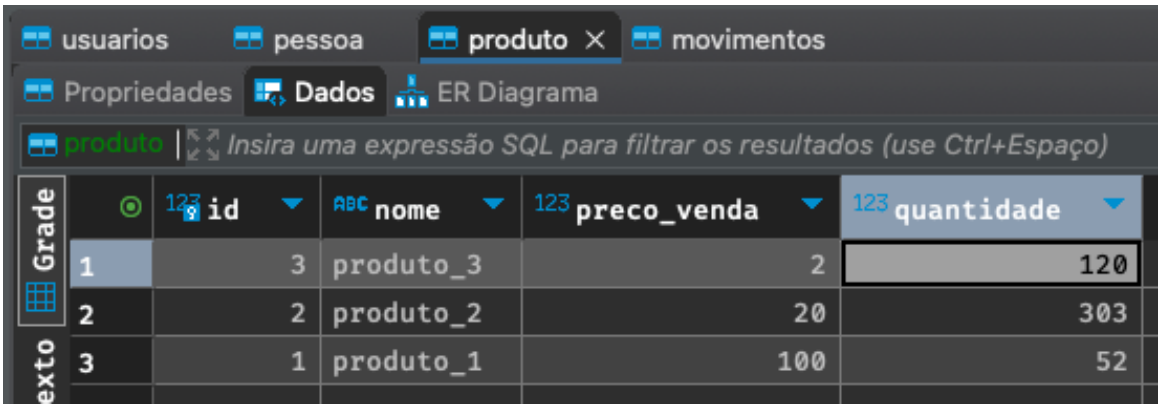
```
remote: - GITHUB PUSH PROTECTION
remote:   ——————————————————————————————————————————————————
remote:     Resolve the following violations before pushing again
remote:
remote:     - Push cannot contain secrets
```

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>cadastroserver.model.Movimentos</class>
 <class>cadastroserver.model.Produto</class>
 <class>cadastroserver.model.Usuarios</class>
 <class>cadastroserver.model.Pessoa</class>
 <properties>
 <property name="javax.persistence.jdbc.url" value="jdbc:postgresql://url_do_meu_server_remoto"/>
 <property name="javax.persistence.jdbc.user" value="faculdade"/>
 <property name="javax.persistence.jdbc.driver" value="org.postgresql.Driver"/>
 <property name="javax.persistence.jdbc.password" value="minha_senha"/>
 </properties>
 </persistence-unit>
</persistence>
```







CadastroClientV2.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Main.java to edit this template
 */
package cadastroclientv2;

import cadastroserver.model.Produto;
import java.io.BufferedReader;
import java.io.EOFException;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.ObjectOutputStream;
import java.io.ObjectInputStream;
import java.net.Socket;
import java.util.List;

/**
 *
 * @author wellingtonfonseca
 */
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("Login: ");
            String login = reader.readLine();
            System.out.print("Senha: ");
            String senha = reader.readLine();
            out.writeObject(login);
            out.writeObject(senha);

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

                if ("X".equals(command)) {
                    break;
                } else if ("L".equals(command)) {
                    List<Produto> produtos = (List<Produto>) in.readObject();
                    for (Produto produto : produtos) {
                        System.out.print(produto.getNome());
                        System.out.print("::");
                        System.out.println(produto.getQuantidade());
                    }
                } else if ("E".equals(command) || "S".equals(command)) {
                    System.out.print("ID Pessoa: ");
                    long idPessoa = Long.parseLong(reader.readLine());
                    out.writeObject(idPessoa);

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

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

                    System.out.print("Valor Unitário: ");
                    float preco = Float.parseFloat(reader.readLine());
                    out.writeObject(preco);
                }
            }
        } catch (EOFException e) {
            // Handle EOFException
            System.out.println("Conexão com o servidor foi encerrada.");
        } catch (IOException | ClassNotFoundException 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.JTextArea;
/**
 *
 * @author wellingtonfonseca
 */
public class SaidaFrame extends JDialog {

    public JTextArea texto;

    public SaidaFrame() {
        texto = new JTextArea();
        texto.setEditable(false);
        this.add(texto);
        this.setBounds(100, 100, 400, 300);
        this.setModal(false);
    }
}
```

ThreadClient.java

```
/*
 * Click nbfs://nbhost/SystemFileSystem/Templates/Licenses/license-default.txt to change this license
 * Click nbfs://nbhost/SystemFileSystem/Templates/Classes/Class.java to edit this template
 */
package cadastroclientv2;
import cadastroserver.model.Produto;
import java.io.IOException;
import javax.swing.JTextArea;
import java.io.ObjectInputStream;
import java.util.List;
import javax.swing.SwingUtilities;

/**
 *
 * @author wellingtonfonseca
 */
public class ThreadClient extends Thread {

    private final ObjectInputStream entrada;
    private final 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();
                SwingUtilities.invokeLater(() -> {
                    if (obj instanceof String) {
                        textArea.append((String) obj + "\n");
                    } else if (obj instanceof List) {
                        @SuppressWarnings("unchecked")
                        List<Produto> produtos = (List<Produto>) obj;
                        for (Produto produto : produtos) {
                            textArea.append(produto.getNome() + " - Quantidade: " + produto.getQuantidade() + "\n");
                        }
                    }
                });
            }
        } catch (IOException | ClassNotFoundException e) {
            e.printStackTrace();
        }
    }
}
```

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 cadastroserver.controller;

import cadastroserver.model.Produto;
import java.util.List;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;

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

    private EntityManagerFactory emf;

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

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

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

    public void edit(Produto produto) throws Exception {
        EntityManager em = getEntityManager();
        try {
            em.getTransaction().begin();
            produto = em.merge(produto);
            em.getTransaction().commit();
        } catch (Exception ex) {
            if (findProduto(produto.getId()) == null) {
                throw new Exception("The produto with id " + produto.getId() + " no longer exists.");
            }
            throw ex;
        } finally {
            em.close();
        }
    }

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

MovimentosJpaController.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.controller;

import cadastroserver.model.Movimentos;
import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import java.io.Serializable;

/**
 *
 * @author wellingtonfonseca
 */
public class MovimentosJpaController {

    private EntityManagerFactory emf;

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

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

    public void create(Movimentos movimento) {
        EntityManager em = getEntityManager();
        try {
            em.getTransaction().begin();
            em.persist(movimento);
            em.getTransaction().commit();
        } 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 cadastroserver.controller;

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

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

    private EntityManagerFactory emf;

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

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

}
```

UsuariosJpaController.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.controller;

import javax.persistence.EntityManager;
import javax.persistence.EntityManagerFactory;
import cadastroserver.model.Usuarios;
import java.util.List;
import javax.persistence.TypedQuery;

/**
 *
 * @author wellingtonfonseca
 */
public class UsuariosJpaController {

 private EntityManagerFactory emf = null;

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

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

 public Usuarios findUsuario(String login, String senha) {
 EntityManager em = getEntityManager();
 try {
 TypedQuery<Usuarios> query = em.createQuery(
 "SELECT u FROM Usuarios u WHERE u.usuario = :usuario AND u.senha = :senha", Usuarios.class);
 query.setParameter("usuario", login);
 query.setParameter("senha", senha);
 List<Usuarios> result = query.getResultList();
 return result.isEmpty() ? null : result.get(0);
 } finally {
 em.close();
 }
 }
}
```