Aluno: Anderson Felipe Felix da Silva

Matrícula: 202401185711

Campus: Boa Viagem, Recife – PE

Missão Prática – Nível 5 – Mundo 3

Objetivo da Prática

Desenvolver uma aplicação em Java utilizando as classes Socket e ServerSocket para comunicação entre cliente e servidor. No lado do servidor, integrar a persistência de dados com JPA e SQL Server. No lado do cliente, exibir os dados obtidos por meio da comunicação com o servidor.

Resultados da Execução

O cliente se conectou com sucesso ao servidor via socket, enviou o comando 'listarPessoas' e recebeu uma lista de objetos Pessoa. Esses dados foram exibidos corretamente no console do cliente.

Análise e Conclusão

1. Como funcionam as classes Socket e ServerSocket?

A classe ServerSocket escuta conexões em uma porta específica. Quando um cliente tenta se conectar, o servidor aceita essa conexão criando um objeto Socket para comunicação. O cliente, por sua vez, utiliza a classe Socket para se conectar ao servidor.

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

As portas são fundamentais para distinguir diferentes serviços em execução em um mesmo host. Elas permitem múltiplas aplicações utilizando a mesma máquina e IP de forma isolada.

3. Para que servem as classes ObjectInputStream e ObjectOutputStream?

São usadas para enviar e receber objetos Java através de conexões. Os objetos devem implementar Serializable para serem transmitidos como bytes.

4. Por que, mesmo utilizando entidades JPA no cliente, foi possível garantir o isolamento do acesso ao banco?

O cliente não acessa diretamente o banco de dados. Ele apenas consome dados enviados pelo servidor, que é o único responsável pela persistência.

Códigos Fonte

Pessoa.java

```
package br.com.estacio.entidade;

import jakarta.persistence.*;

@Entity
@Table(name = "Pessoa")
public class Pessoa {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long id;
   private String nome;

   public Long getId() { return id; }
   public void setId(Long id) { this.id = id; }
   public String getNome() { return nome; }
   public void setNome(String nome) { this.nome = nome; }
}
```

ClienteSocket.java

```
package br.com.estacio.socket;

import java.io.ObjectInputStream;
import java.io.ObjectOutputStream;
import java.net.Socket;
import java.util.List;
import br.com.estacio.entidade.Pessoa;

public class ClienteSocket {
   public static void main(String[] args) {
      String host = "localhost";
```

```
        int porta = 12345;

        try (
            Socket socket = new Socket(host, porta);
            ObjectOutputStream saida = new ObjectOutputStream(socket.getOutputStream());
            ObjectInputStream entrada = new ObjectInputStream(socket.getInputStream());
        ) {
            String comando = "listarPessoas";
            saida.writeObject(comando);

            Object resposta = entrada.readObject();

            if (resposta instanceof List) {
                List<Pessoa> pessoas = (List<Pessoa>) resposta;
                System.out.println("Pessoas recebidas do servidor:");
                for (Pessoa p : pessoas) {
                    System.out.println("ID: " + p.getId() + " Nome: " + p.getNome());
                }
            } else {
                System.out.println("Resposta do servidor: " + resposta);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

App.java

```
package br.com.estacio;

import br.com.estacio.socket.ServidorSocket;
import jakarta.persistence.EntityManagerFactory;
import jakarta.persistence.Persistence;

public class App {
    public static void main(String[] args) {
        EntityManagerFactory emf = Persistence.createEntityManagerFactory("cadastroPU");
        ServidorSocket servidor = new ServidorSocket(emf);
        servidor.iniciar();
    }
}
```