



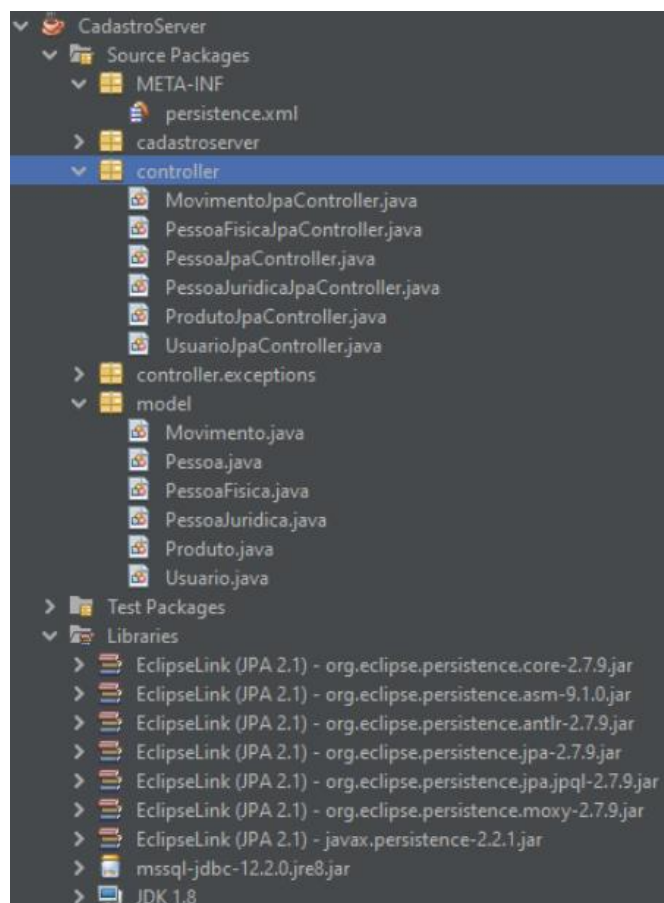
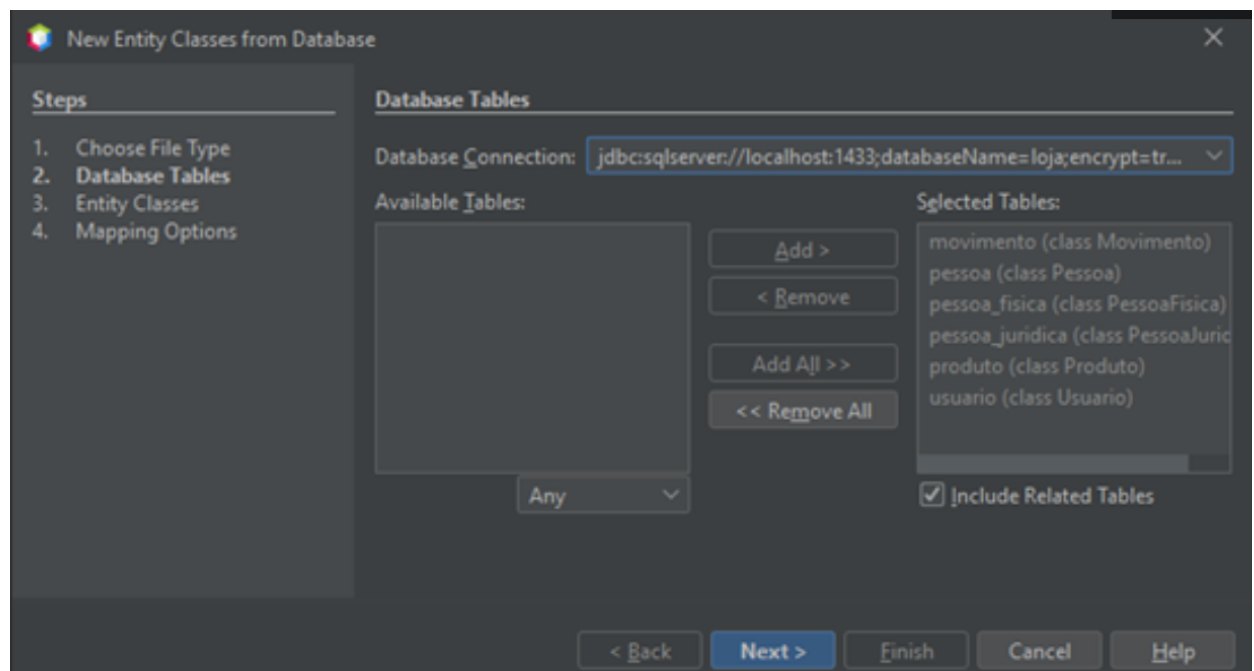
# Estácio

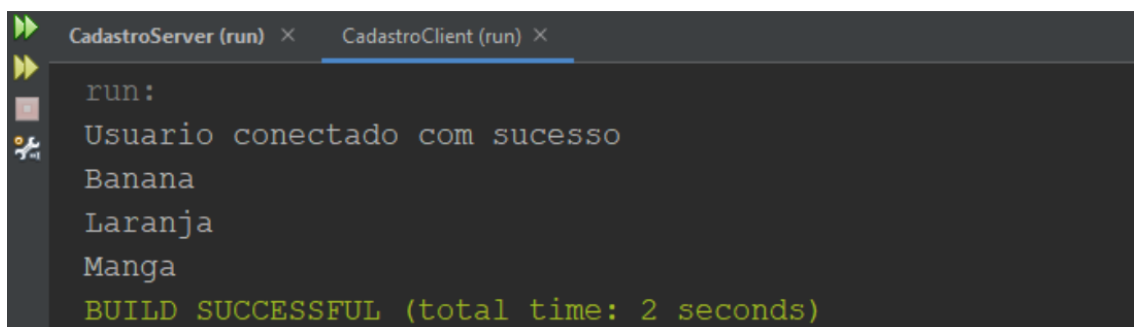
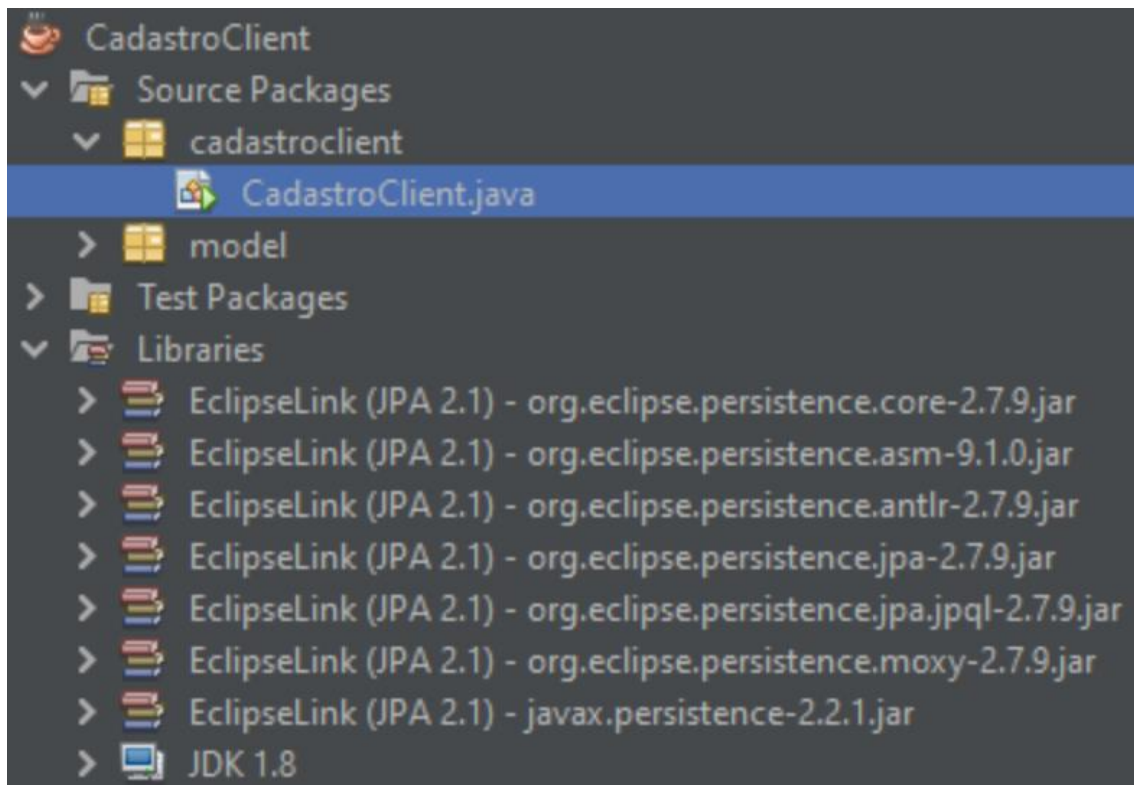
UNIVERSIDADE ESTÁCIO DE SÁ POLO IBIRITÉ  
CURSO: DESENVOLVIMENTO FULL STACK  
DISCIPLINA: POR QUE NÃO PARALELIZAR?

EVERTON GOMES COSTA  
TURMA: 22.3  
3º SEMESTRE

IBIRITÉ-MG  
2023

# 1º procedimento





## Análise e Conclusão:

### 1. Como funcionam as classes Socket e ServerSocket?

Socket e ServerSocket são classes em Java usadas para comunicação em rede. Socket: Representa o ponto final de uma conexão de rede e permite a comunicação bidirecional entre programas. Um Socket pode ser tanto um cliente quanto um servidor. ServerSocket: É usado por servidores para aguardar e aceitar conexões de clientes. Uma vez que uma conexão é aceita, um novo Socket é criado para lidar com a comunicação com o cliente.

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

As portas são números de identificação associados a processos em execução em um host. Eles são usados para direcionar o tráfego de rede para aplicativos específicos. Em uma conexão, a combinação de IP e porta identifica exclusivamente um serviço em um host. Portas bem conhecidas são atribuídas a serviços específicos (por exemplo, HTTP usa a porta 80). A concepção da porta é essencial para direcionar as comunicações para o serviço desejado.

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

**ObjectInputStream** e **ObjectOutputStream** são usados para realizar entrada e saída de objetos em Java.

**ObjectOutputStream**: Converte objetos em fluxos de bytes para serem enviados pela rede ou salvos em arquivos.

**ObjectInputStream**: Lê esses fluxos de bytes e os converte de volta em objetos.

Os objetos transmitidos devem ser serializáveis para que possam ser convertidos em bytes e reconstruídos. A serialização é o processo de conversão de um objeto em uma sequência de bytes.

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

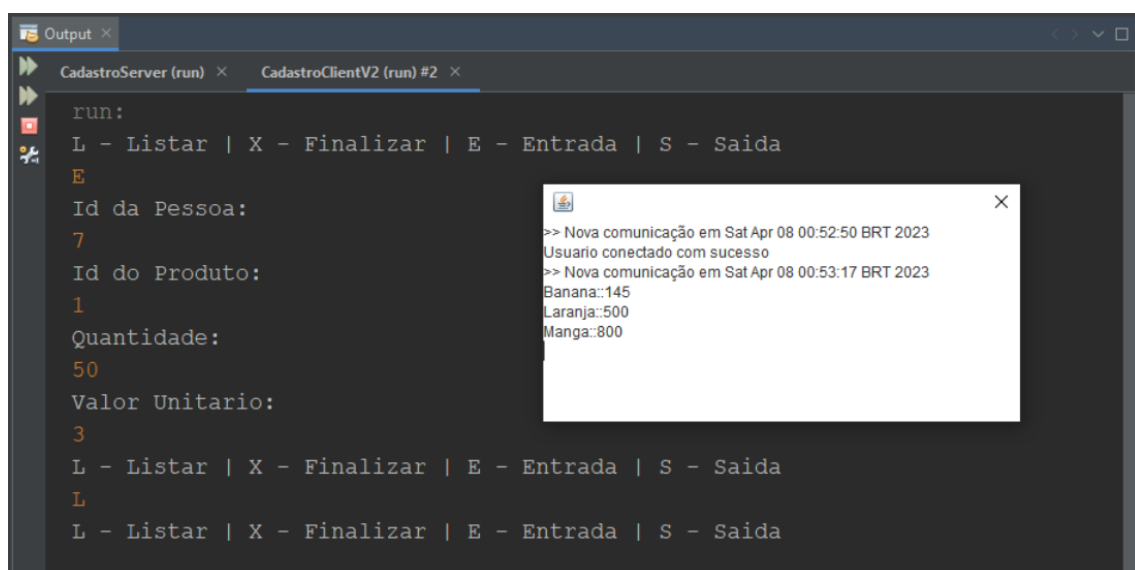
O isolamento do acesso ao banco de dados é alcançado por meio da arquitetura de camadas e da utilização de JPA.

As classes de entidades JPA representam entidades do banco de dados e contêm apenas a lógica de mapeamento objeto-relacional. Eles não contêm lógica de acesso direto ao banco de dados.

No cliente, as operações de leitura e escrita são realizadas usando métodos de camada de serviço, que encapsulam as operações do banco de dados. O JPA gerencia a comunicação com o banco de dados, fornecendo um meio de abstração.

Essa abordagem permite a manutenção do isolamento, uma vez que o acesso direto ao banco de dados é tratado apenas pela camada de serviço no servidor, enquanto o acesso ao cliente apenas com as classes de entidade e serviços JPA.

## 2º procedimento



```
run:
L - Listar | X - Finalizar | E - Entrada | S - Saida
E
Id da Pessoa:
7
Id do Produto:
1
Quantidade:
50
Valor Unitario:
3
L - Listar | X - Finalizar | E - Entrada | S - Saida
L
L - Listar | X - Finalizar | E - Entrada | S - Saida
```

>> Nova comunicação em Sat Apr 08 00:52:50 BRT 2023  
Usuario conectado com sucesso  
>> Nova comunicação em Sat Apr 08 00:53:17 BRT 2023  
Banana::145  
Laranja::500  
Manga::800

## **Análise e Conclusão:**

### **1. Como as Threads podem ser utilizadas para o tratamento assíncrono das respostas enviadas pelo servidor?**

Threads podem ser usados para tratar respostas assíncronas do servidor permitindo que o programa continue a execução enquanto aguarda uma resposta.

Ao receber uma resposta do servidor em um thread separado, o programa principal não fica bloqueado, podendo realizar outras tarefas.

Isso é particularmente útil em interfaces gráficas de usuário (GUIs), onde a resposta do servidor não deve congelar a interação do usuário com a aplicação.

### **2. Para que serve o método invokeLater, da classe SwingUtilities?**

O método invokeLater da classe SwingUtilities é usado para executar uma ação na thread de despacho de eventos do Swing (também conhecida como EDT - Event Dispatch Thread).

Em aplicações Swing, todas as interações com componentes visuais devem ocorrer na EDT para garantir a consistência e a segurança.

invokeLater é usado para empacotar uma tarefa (Runnable) que será realizada na EDT, permitindo a execução de operações gráficas de forma assíncrona.

### **3. Como os objetos são enviados e recebidos pelo Socket Java?**

Para enviar e receber objetos pelo Socket em Java, é comum usar ObjectOutputStream para enviar objetos e ObjectInputStream para recebê-los.

A serialização é utilizada para converter objetos em bytes para transmissão pela rede.

O lado receptor usa ObjectInputStream para reconstruir os objetos a partir dos bytes recebidos.

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

Comportamento Síncrono:

Em operações síncronas, o cliente aguarda uma resposta do servidor antes de continuar a execução. Isso pode resultar em bloqueio, especialmente em operações que desativam o tempo de resposta do servidor. Pode ser mais simples de implementar, pois as operações ocorrem sequencialmente.

Comportamento Assíncrono:

Em operações assíncronas, o cliente pode continuar a execução enquanto aguarda a resposta do servidor. Permite um melhor aproveitamento dos recursos do cliente, já que ele não fica inativo enquanto aguarda as respostas. Pode ser mais complexo de implementação devido à necessidade de lidar com eventos assíncronos e possíveis problemas de concorrência.

A escolha entre comportamento síncrono e assíncrono depende dos requisitos específicos da aplicação e das preferências de design. O comportamento assíncrono é geralmente preferido em situações em que a responsividade e a eficiência são críticas.