



# Estácio

**Campus:** Conceição - São Paulo/SP

**GRADUAÇÃO DESENVOLVIMENTO FULL STACK**

**Disciplina:** Nível 5 - Por que não paralelizar

**Turma:** 2022.03 - Mundo 3

**Aluna:** Fernanda G. Vargas

**Matrícula:** 202208836305



## 1º Procedimento | Criando o Servidor e Cliente de Teste

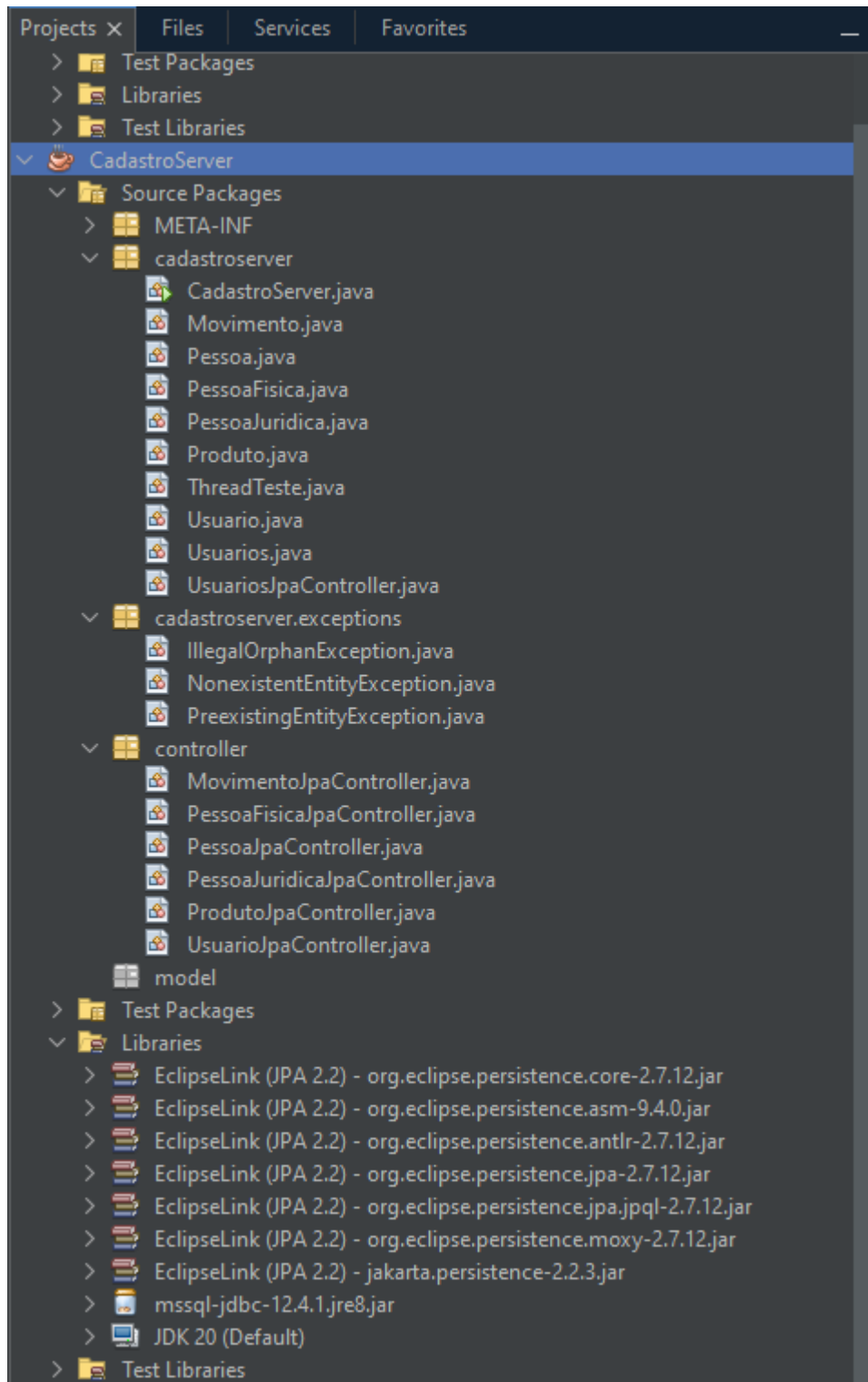
**1- Título da Prática:** Criando o Servidor e Cliente de Teste

**2- 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.

**3- Todos os códigos solicitados neste roteiro de aula:**

Disponível: <https://github.com/FerGVargas>

**4- Os resultados da execução dos códigos também devem ser apresentados:**



## 5- Análise e Conclusão:

### 5.a- Como funcionam as classes Socket e ServerSocket?

São utilizadas em conjunto para estabelecer a comunicação entre um servidor e seus clientes. O ServerSocket aguarda por conexões e cria um Socket para cada cliente que se conecta.

### 5.b- Qual a importância das portas para a conexão com servidores?

Comunicação entre computadores em uma rede, rede entre servidores e clientes. permite que vários serviços estejam em execução simultaneamente no mesmo servidor. Cada serviço pode ser acessado através de sua porta exclusiva,

possibilitando a concorrência e o compartilhamento eficiente dos recursos do servidor. As portas são fundamentais para a identificação, roteamento e isolamento eficientes de serviços em uma rede. Elas desempenham um papel essencial na comunicação cliente-servidor, permitindo que diferentes serviços coexistam e sejam acessados de maneira ordenada e segura.

**5.c- Para que servem as classes de entrada e saída `ObjectInputStream` e `ObjectOutputStream`? São utilizadas para realizar a serialização e desserialização de objetos. E por que os objetos transmitidos devem ser serializáveis?** Para transmitir objetos entre diferentes máquinas ou persistir objetos em armazenamento. Ao serializar um objeto, ele pode ser transformado em uma sequência de bytes que pode ser transmitida ou armazenada.

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

Classes de entidades JPA torna possível interação do cliente com dados de forma orientada a objetos, enquanto a lógica de acesso ao BD é tratada no servidor. O isolamento no acesso ao BD é possível usando com padrão arquitetura Cliente-Servidor e a abstração pelas classes de entidades JPA.

## 2º Procedimento | Servidor Completo e Cliente Assíncrono

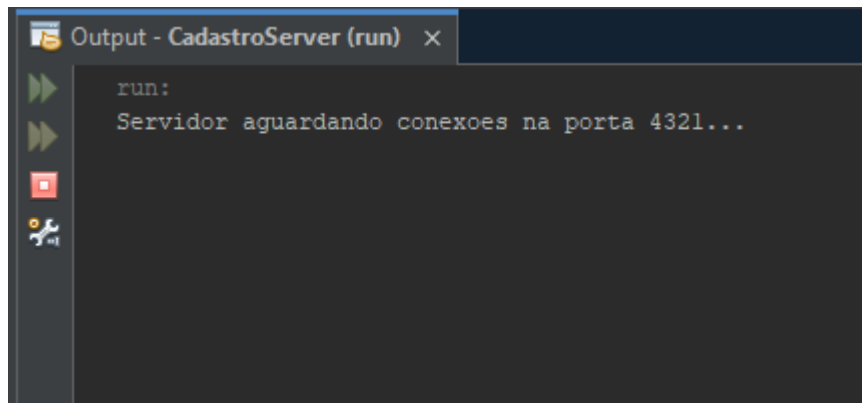
**1- Título da Prática: Por que não paralelizar**

**2- 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.

**3- Todos os códigos solicitados neste roteiro de aula:**

Disponível: <https://github.com/FerGVargas>

**4- Os resultados da execução dos códigos também devem ser apresentados:**



```
run:
Servidor aguardando conexoes na porta 4321...
```

## Análise e Conclusão:

### 5.a- Como as Threads podem ser utilizadas para o tratamento assíncrono das respostas enviadas pelo servidor?

Permitindo que operações demoradas ou bloqueantes sejam realizadas sem interromper a execução principal do programa. Existem várias maneiras de implementar esse tratamento assíncrono, e uma abordagem comum é o uso de threads ou tarefas assíncronas.

### 5.b- Para que serve o método `invokeLater`, da classe `SwingUtilities`?

Para executar uma determinada tarefa de forma assíncrona no Event Dispatch Thread 'EDT'. O EDT é a thread principal responsável por manipular a interface gráfica em aplicações Swing. Ao usar `invokeLater`, você garante que a tarefa seja executada de maneira segura no contexto da EDT.

### 5.c- Como os objetos são enviados e recebidos pelo Socket Java?

A serialização converte os objetos em uma sequência de bytes que podem ser transmitidos pela rede. O `ObjectOutputStream` é usado para escrever objetos em bytes, que são enviados pelo socket. O `ObjectInputStream` é usado para ler os bytes recebidos e reconstruir os objetos originais. A classe dos objetos deve implementar a interface `Serializable` para permitir a serialização e desserialização.

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

Com Sockets Java, continua realizando outras tarefas enquanto esperam por respostas do servidor, porém é necessário retorno do cliente para o servidor continuar, o que faz menos eficiente. O comportamento assíncrono garante a fluidez na interação do usuário e otimizar o uso de recursos do sistema.