

Projeto Final – Sistema Distribuído

Brendon Vicente Rocha Silva (18202189), Breno de Brida (18204020) e
Vitor Philip Starucka (19200442)

December 2022

1 Proposta do projeto

Projetar e desenvolver um sistema distribuído combinando diferentes tecnologias de suporte a computação distribuída.

1.1 Ideia

Sistema distribuído de controle de estoque.

1.1.1 Capacidades e uso

A ideia do software é facilitar o gerenciamento de um estoque que pode ser acessado por múltiplos usuários simultaneamente. Ele foi feito de forma genérica, dessa forma, é possível controlar o estoque de quaisquer tipos de produtos que os usuários vejam utilidade em cadastrar. Poderia servir por exemplo, para o controle das vendas e entrada de estoque de um restaurante fast-food.

1.1.2 Execução

Para rodar o sistema, posicionado na pasta do projeto, é só executar em seu terminal o arquivo:

```
compile.sh
```

Então rodar o servidor:

```
server.sh
```

E por fim, quantas instâncias de clientes quiser, com o arquivo:

```
client.sh
```

Também é possível executar diretamente o servidor e os clientes através da sua IDE de preferência. Configurando o PATH para o .jar "json-simple-1.1.1.jar" localizado em: Sistema-Estoque-Distribuido/src/resources

1.1.3 Demonstração de uso

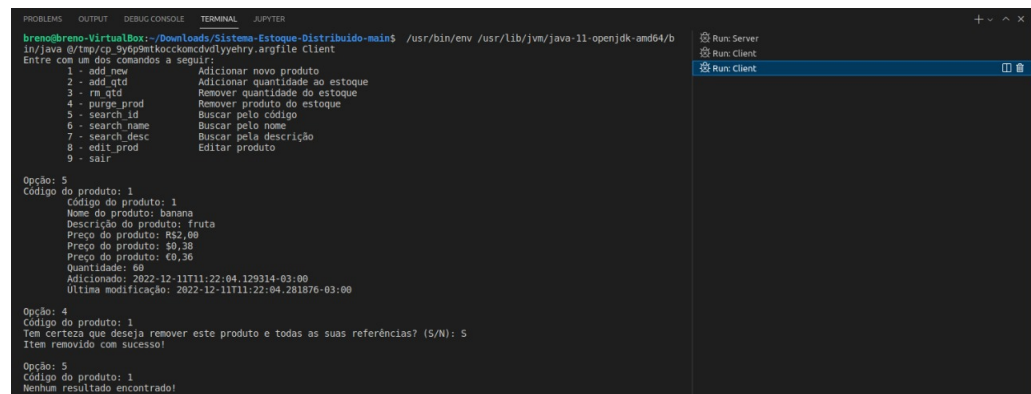
Com nosso servidor ativo, utilizamos a opção de inserir um produto e carregamos as informações para que ele seja adicionado.



```
breng@breng-VirtualBox:~/Downloads/Sistema-Estoque-Distribuido-main$ ./usr/bin/env ./usr/lib/jvm/java-11-openjdk-amd64/bin/java @/tmp/cp.5y6p9mtkocckoncdvlyyehry.argfile client
breng@breng-VirtualBox:~/Downloads/Sistema-Estoque-Distribuido-main$ ./usr/bin/env ./usr/lib/jvm/java-11-openjdk-amd64/bin/java @/tmp/cp.5y6p9mtkocckoncdvlyyehry.argfile client
Entre com um dos comandos a seguir:
1 - add_new      Adicionar novo produto
2 - add_qtd      Adicionar quantidade ao estoque
3 - rm_qtd       Remover quantidade do estoque
4 - purge_prod   Remover produto do estoque
5 - search_id    Buscar pelo código
6 - search_name  Buscar pelo nome
7 - search_desc  Buscar pela descrição
8 - edit_prod    Editar produto
9 - sair

Opção: 1
Nome do produto: banana
Descrição do produto: fruta
Preço do produto: R$2
Quantidade no estoque: 60
Item adicionado com sucesso!
Código do produto: 1
```

Através de outra instância do cliente, utilizamos a opção para resgatar os dados de um determinado produto já cadastrado. Então realizamos as operações 4 e 5, para remover aquele produto e verificar o sucesso de sua remoção.



```
breng@breng-VirtualBox:~/Downloads/Sistema-Estoque-Distribuido-main$ ./usr/bin/env ./usr/lib/jvm/java-11-openjdk-amd64/bin/java @/tmp/cp.5y6p9mtkocckoncdvlyyehry.argfile client
breng@breng-VirtualBox:~/Downloads/Sistema-Estoque-Distribuido-main$ ./usr/bin/env ./usr/lib/jvm/java-11-openjdk-amd64/bin/java @/tmp/cp.5y6p9mtkocckoncdvlyyehry.argfile client
Entre com um dos comandos a seguir:
1 - add_new      Adicionar novo produto
2 - add_qtd      Adicionar quantidade ao estoque
3 - rm_qtd       Remover quantidade do estoque
4 - purge_prod   Remover produto do estoque
5 - search_id    Buscar pelo código
6 - search_name  Buscar pelo nome
7 - search_desc  Buscar pela descrição
8 - edit_prod    Editar produto
9 - sair

Opção: 5
Código do produto: 1
Código do produto: 1
Nome do produto: banana
Descrição do produto: fruta
Preço do produto: R$2,00
Preço do produto: $0,38
Preço do produto: €0,36
Quantidade: 60
Adicionado: 2022-12-11T11:22:04.129314-03:00
Última modificação: 2022-12-11T11:22:04.281876-03:00

Opção: 4
Código do produto: 1
Tem certeza que deseja remover este produto e todas as suas referências? (S/N): S
Item removido com sucesso!

Opção: 5
Código do produto: 1
Nenhum resultado encontrado!
```

Observe que: ao resgatar os dados de um produto, retornamos informações que não haviam sido informadas pelo usuário, como preço do produto em dólar e em euro, assim como a data exata em que o mesmo foi adicionado e modificado. Essas informações são resgatadas por Web Services, e o usuário não sabe dizer se são informações resgatadas localmente ou remotamente.

2 Arquitetura

Pensando nas demandas da ideia proposta, com os conhecimentos agregados em aula e os requisitos do professor para o projeto, foi desenvolvido um sistema Cliente/Servidor, utilizando Java RMI e Web Services, consultando APIs rest. Visando e cumprindo, a implementação de um sistema completo, mas descomplicado.

2.1 Requisitos tecnológicos

- **Heterogeneidade:** Permite que nodos/componentes com diferentes propriedades façam parte de um mesmo sistema: Web Services se comunicam com o sistema.
- **Escalabilidade:** O número de usuários e a quantidade de recursos pode aumentar sem limitações definidas. Para diminuir a latência, a transferência de dados entre requisitante e servidor é minimizada, de modo que o formulário de adição de um produto capta todas as informações de uma única vez, ao invés de ter um envio por partes.

2.2 Transparências

O projeto proposto conta com as seguintes transparências:

- **Transparência de acesso:** recursos locais e remotos são acessados pelas mesmas operações. De forma que, quando o cliente recebe a informação sobre seus produtos, para ele não há diferenciação entre o que está presente localmente e o que é resgatado através dos Web Services.
- **Transparência de localização:** recursos são acessados sem que sua localização seja determinada explicitamente. Não há apresentação ao usuário de informações que digam a ele onde os recursos estão localizados.
- **Transparência de concorrência:** processos executam concorrentemente, utilizando recursos compartilhados, sem interferirem na execução dos outros. Múltiplas instâncias de clientes poderiam por exemplo, resgatar as informação sobre o estoque, sem serem afetados.

2.3 Diagrama da arquitetura

Por fim, o seguinte diagrama apresenta a estrutura do sistema implementado:

