

**UNIVERSIDADE FEDERAL DE SANTA CATARINA – UFSC
CENTRO TECNOLÓGICO DA UFSC
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO**

RELATÓRIO T2 - Comunicação em Grupo

**LUIZ OTÁVIO SANTOS REIS (19102925)
THIAGO ZIMMERMANN LOUREIRO CHAVES (19100547)**

**Disciplina: Computação Distribuída
Professor: Odorico Machado Mendizabal**

FLORIANÓPOLIS, 11 DE MARÇO DE 2022

Luiz Otávio Santos Reis
Thiago Zimmermann Loureiro Chaves

RELATÓRIO DE COMPUTAÇÃO DISTRIBUÍDA

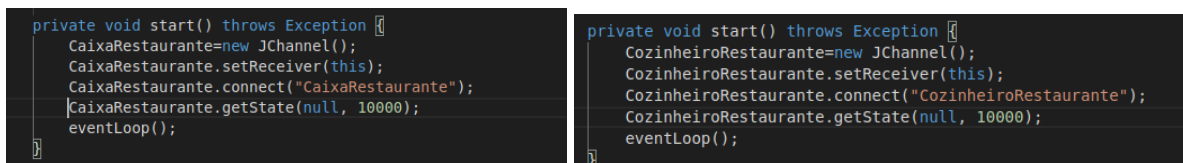
Florianópolis
2022

1) Aplicação Desenvolvida

Neste trabalho foi proposta a criação de uma aplicação distribuída utilizando **Java** e **JGroups** como suporte a comunicação. Nesse sentido, criamos um sistema de restaurante chamado “**Md Conald’s**” onde existe a aplicação caixa e cozinheiro, onde ambos possuem um canal de comunicação próprio, juntamente com o restaurante que possui conexão com os 2. Além disso, criamos uma **interface gráfica** básica com o auxílio da aplicação **NetBeans**.

2) Principais Decisões para Implementação

- Criamos dois tipos de usuário: o caixa e o cozinheiro onde vão executar em terminais distintos onde cada um se comunicará através de seu canal único que apenas o restaurante terá acesso.
- Desenvolvemos a interface gráfica através do **NetBeans** uma vez que seria uma interface simples de se utilizar para ambos que nunca haviam usado java.
- Para manter o histórico de pedidos utilizamos um txt chamado de “**pedido.txt**”.



```
private void start() throws Exception {
    CaixaRestaurante=new JChannel();
    CaixaRestaurante.setReceiver(this);
    CaixaRestaurante.connect("CaixaRestaurante");
    CaixaRestaurante.getState(null, 10000);
    eventLoop();
}

private void start() throws Exception {
    CozinheiroRestaurante=new JChannel();
    CozinheiroRestaurante.setReceiver(this);
    CozinheiroRestaurante.connect("CozinheiroRestaurante");
    CozinheiroRestaurante.getState(null, 10000);
    eventLoop();
}
```

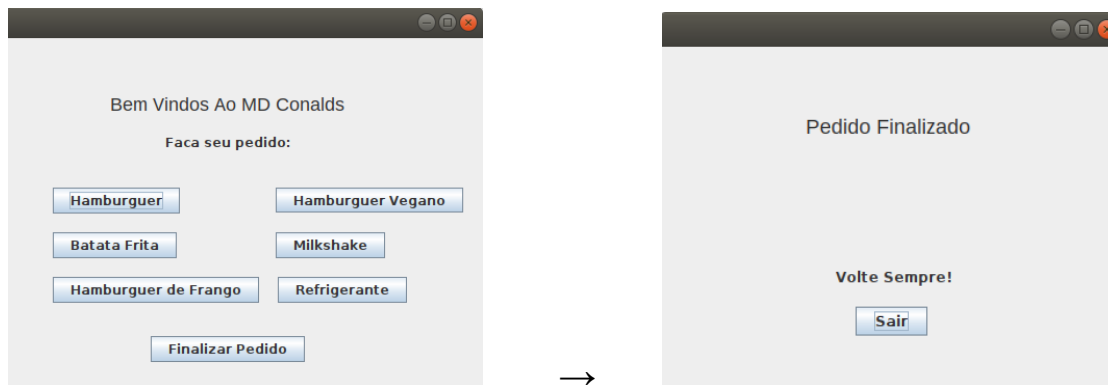
Figura 1 e 2: Figura a esquerda é iniciação do canal do caixa e da direita do cozinheiro



```
private void start() throws Exception {
    /* Caixa Restaurante */
    CaixaRestaurante=new JChannel();
    CaixaRestaurante.setReceiver(this);
    CaixaRestaurante.connect("CaixaRestaurante");
    CaixaRestaurante.getState(null, 10000);
    /* Cozinheiro Restaurante */
    CozinheiroRestaurante=new JChannel();
    CozinheiroRestaurante.setReceiver(this);
    CozinheiroRestaurante.connect("CozinheiroRestaurante");
    CozinheiroRestaurante.getState(null, 10000);
    eventLoop();
}
```

Figura 3: Configuração dos dois jchannels no Restaurante.java

- Nesse contexto, criamos a interface gráfica para os dois usuarios: Caixa e o Cozinheiro



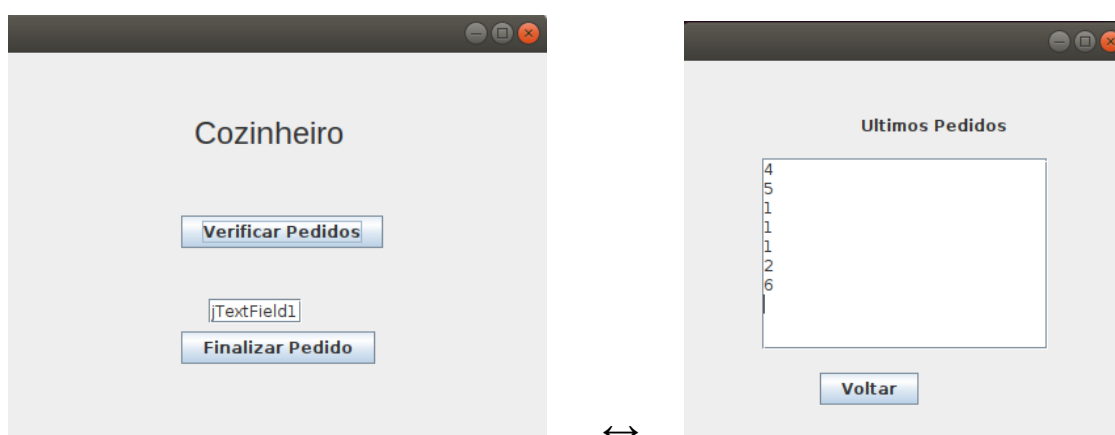
Telas do caixa: Transiciona para o segunda através do “finalizar pedido”

```
thiago@thiago-VirtualBox: ~/Downloads/DistribuidaFinal/T2
Arquivo Editar Ver Pesquisar Terminal Ajuda
ADVERTÊNCIA: receive buffer of socket java.net.DatagramSocket@67784306 was set to 20MB, but the OS only allocated 212,99KB. This might lead to performance problems. Please set your max receive buffer in the OS correctly (e.g. net.core.rmem_max on Linux)
mar 11, 2022 4:57:19 PM org.jgroups.logging.JDKLogImpl warn
ADVERTÊNCIA: send buffer of socket java.net.MulticastSocket@210366b4 was set to 640KB, but the OS only allocated 212,99KB. This might lead to performance problems. Please set your max send buffer in the OS correctly (e.g. net.core.wmem_max on Linux)
mar 11, 2022 4:57:19 PM org.jgroups.logging.JDKLogImpl warn
ADVERTÊNCIA: receive buffer of socket java.net.MulticastSocket@210366b4 was set to 25MB, but the OS only allocated 212,99KB. This might lead to performance problems. Please set your max receive buffer in the OS correctly (e.g. net.core.rmem_max on Linux)

-----
GMS: address=thiago-VirtualBox-19571, cluster=CaixaRestaurante, physical address=fe80:0:0:0:ff62:cc84:7897:1b12%2:57989
-----

# Aguarde o cliente efetuar um pedido #
0 pedido: Hamburguer de ID 1 foi encaminhado aos cozinheiros
0 pedido: Batata Frita de ID 2 foi encaminhado aos cozinheiros
0 pedido: Refrigerante de ID 6 foi encaminhado aos cozinheiros
```

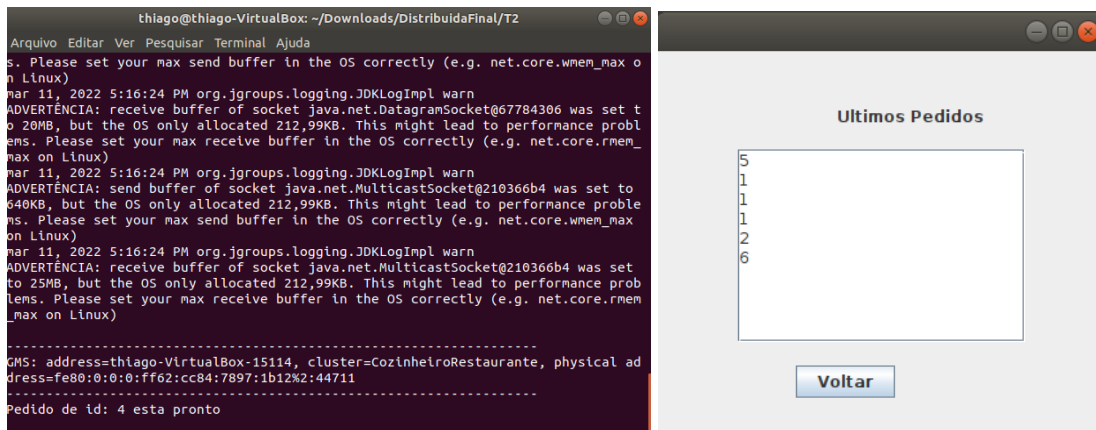
Ao clicar no botão dos pedidos ele encaminha para o **JChannel CaixaRestaurante** e escreve no **pedidos.txt** para manter um histórico de pedidos.



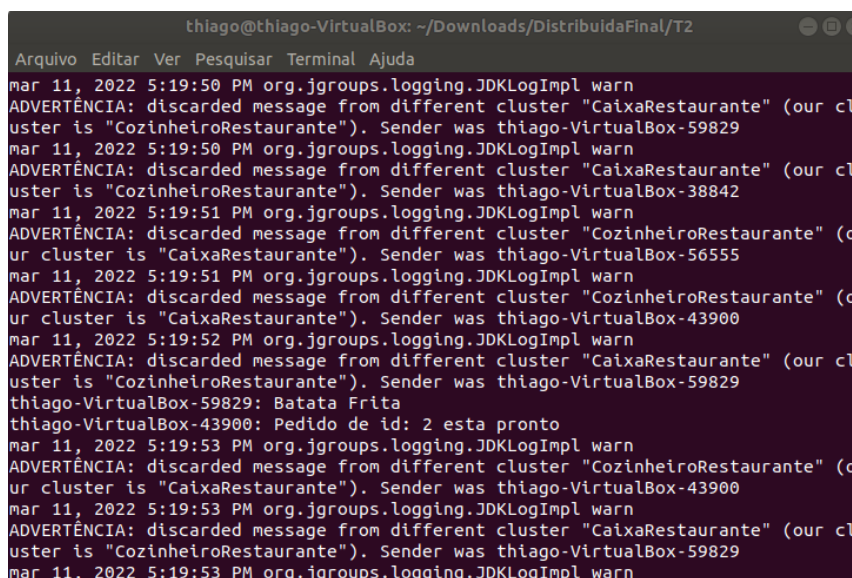
Telas do Cozinheiro: Transiciona para o segunda através do “verificar pedidos” e volta pelo botão “voltar”

Ao preenchermos o campo **jtextField1** com o pedido enviamos a mensagem pelo

Jchannel CozinheiroRestaurante e retiramos o pedido do **pedidos.txt**.



Resultado da entrega de um pedido por parte do cozinheiro.



Tela do Restaurante.java demonstrando a entrega de mensagens de ambos os canais.

3) Problemas Encontrados

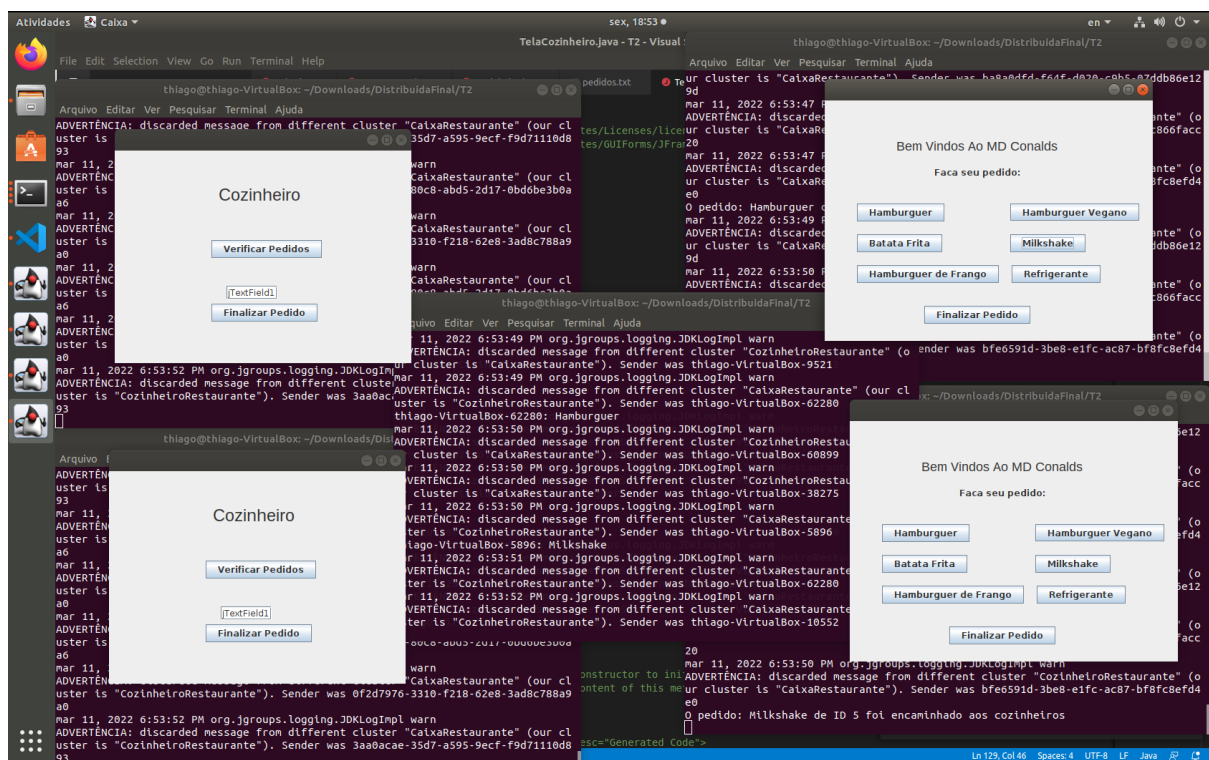
- Baixo conhecimento inicial em Java.
- Desconhecimento sobre o JGroups.
- Pouca documentação sobre o JGroups.
- Retirar os warnings na utilização de dois Jchannels, conversamos com o professor e olhamos diversos fóruns mas não achamos um jeito de configurar. (O arquivo solicitado para mudança não existia na nossa versão do JGroups).

4) Estudo de Caso da Aplicação

Testamos nossa aplicação em diversos casos como:

- 1 Caixa, 1 Restaurante.
- 1 Cozinheiro, 1 Restaurante.
- 1 Caixa, 1 Cozinheiro e 1 Restaurante.
- 2 Caixas, 2 Cozinheiros e 1 Restaurante.
- 3 Caixas, 3 Cozinheiros e 1 Restaurante.

Nesse sentido, acreditamos que funcionaria para “n” ou mais casos. Além disso, garantimos a não concorrência dos caixas pela utilização de um histórico de pedidos realizados em um txt, que é atualizado frequentemente.



Exemplo da realização 2 Cozinheiros, 2 Caixas e 1 Restaurante.

5) Como Executar (ReadMe do Projeto)

- Download do JGroups. Obs: nosso projeto já vem com o JGroups 3.0.0 dentro do diretório.
- `javac *.java`
- Abra um terminal e digite: `java Restaurante`
- Abra um terminal e digite: `java Caixa`
- Abra um terminal e digite: `java Cozinheiro`
- No terminal do caixa faça um pedido clicando nos botões
- No terminal do cozinheiro clique no botão “verificar pedidos” e clique no botão “voltar”.
- Digite no campo o pedido que deseja retirar.
- Clique no botão “verificar pedidos” novamente, e verificará que o pedido não estará mais lá.
- Essas operações de registrar pedido e concluir pedido serão registradas no `Restaurante.java`, no entanto entre os warnings.

6. Bibliografia

<http://www.jgroups.org/tutorial/html/ch01.html>

<http://www.jgroups.org/tutorial/html/ch02.html>

https://www.tutorialspoint.com/java/java_basic_syntax.htm