

# Computação Distribuída: Laboratório 02

Rafael Begnini de Castilhos (20205642)

31 de outubro de 2022

## 1 Relatório

### 1.1 Instruções:

Para compilar e executar, é necessário:

- Mover para o diretório source: *cd src*
- Compilar: *./compile*
- Em um terminal, executar o servidor: *./server*
- Em um ou vários terminais, executar os clientes: *./client*

### 1.2 Execução:

A sequência de comandos executadas é a seguinte, e seus respectivos retornos estão na Figura 1.

- Cliente 1: Adiciona "aaa" no índice 0. Adiciona "bbb" no índice 1.
- Cliente 2: Captura tamanho. Captura elemento no índice 0.
- Cliente 3: Atualiza índice 1 com "zzz". Remove índice 0.
- Cliente 4: Captura elemento no índice 1. Captura elemento no índice 0. Captura tamanho.
- Cliente 3: Adiciona "ccc" no índice 2.

- Cliente 2: Captura elemento no índice 2.
- Cliente 4: Cliente 1: Remove índice 1.
- Cliente 2: Captura tamanho.
- Cliente 4: Captura elemento no índice 2.

```

sparkdeveloper@SPK-NOT-0022: ~/Downloads/rmi/src
rmiregistry em execucao. Reiniciando o rmiregistry
iniciando o servidor

Cliente 1
-> add <index> <element>
-> get <index>
-> remove <index>
-> size
-> exit
add 0 aaa
Elemento adicionado
add 1 bbb
Elemento adicionado
remove 1
Elemento removido: zzz

Cliente 2
-> size
-> exit
size
Tamanho: 2
get 0
Elemento capturado: aaa
get 2
Elemento capturado: ccc
size
Tamanho: 1

Cliente 3
Digite uma operação:
-> add <index> <element>
-> get <index>
-> remove <index>
-> size
-> exit
add 1 zzz
Elemento adicionado
remove 0
Elemento removido: aaa
add 2 ccc
Elemento adicionado

Cliente 4
Digite uma operação:
-> add <index> <element>
-> get <index>
-> remove <index>
-> size
-> exit
get 1
Elemento capturado: zzz
get 0
Elemento capturado: null
size
Tamanho: 1
get 2
Elemento capturado: ccc

```

Figura 1: Execução de 4 clientes.

### 1.3 Consideração execução concorrente:

O acesso concorrente implicaria em uma condição de corrida, e é necessário gerenciar do uso da memória compartilhada a fim de que um processo não interfira na execução de outro processo sendo executado concorrentemente. Por isso, utilizando *synchronized* na assinatura dos métodos, é possível monitorar um objeto que garante exclusão mútua na execução de procedimentos a ele associados.