

Pró-Reitoria Acadêmica
Curso de Bacharelado em Ciência da Computação
Trabalho da Disciplina Programação concorrente e
distribuída.

Sistema de reserva de assentos

Autores:
Ian Araújo Caldas;
Joyce Aline Martins Fonseca;
Júlia Gabriela Gomes Da Silva;
Lara Ewellen De Carvalho Rocha;
Luiza Lima De Sá.
Orientador: Prof. MARCELO EUSTAQUIO SOARES DE
LIMA JUNIOR

Brasília - DF
2025

**IAN ARAÚJO CALDAS;
JOYCE ALINE MARTINS FONSECA;
JÚLIA GABRIELA GOMES DA SILVA;
LARA EWELLEN DE CARVALHO ROCHA;
LUIZA LIMA DE SÁ.**







SISTEMA DE RESERVA DE ASSENTOS

Documento apresentado ao Curso de graduação de Bacharelado em Ciência da Computação da Universidade Católica de Brasília, como requisito parcial para obtenção da aprovação na disciplina de Programação concorrente e distribuída.

**Orientador: Prof. MARCELO
EUSTAQUIO SOARES DE LIMA
JUNIOR**

**Brasília
2025**

SUMÁRIO

 Sistema de Reserva de Assentos	4
 Objetivo.....	4
 Problemas Resolvidos com Sincronização.....	4
 Como funciona.....	4
♦ Estrutura.....	4
 Instruções de Uso.....	5
Execução.....	5
Exemplo de uso:.....	5
 Sincronização.....	6

📋 Sistema de Reserva de Assentos

📌 Objetivo

Desenvolver um sistema de reserva de assentos online para um cinema com 100 assentos numerados de 1 a 100. O sistema deve permitir múltiplos acessos simultâneos por clientes, com possibilidade de reservar um ou mais assentos por vez. A comunicação é feita via **sockets**, com dois módulos: **Servidor** e **Cliente**.

🔒 Problemas Resolvidos com Sincronização

Sem controle adequado, o sistema estaria sujeito a **condições de corrida** (race conditions), como:

- ❌ **Reservas duplicadas** (dois clientes reservando o mesmo assento ao mesmo tempo)
- ❌ **Estado inconsistente** (o sistema mostra o assento como disponível, mas ele já foi reservado)
- ❌ **Perda de atualizações** (reservas sobrescrevendo umas às outras)

✅ Para resolver isso, foi usado um **mutex (lock)** com a biblioteca **threading** do Python. Assim, **somente uma thread por vez pode acessar ou modificar a lista de assentos**.

🧠 Como funciona

♦ Estrutura

- **Servidor (`servidor.py`):**
 - Controla a lista de assentos.
 - Garante exclusão mútua com `threading.Lock()`.
 - Exibe todas as operações no terminal com `print()`.
 - Atende múltiplos clientes usando `threading.Thread`.

- **Cliente (`cliente.py`):**
 - Conecta ao servidor via socket.
 - Envia comandos para ver ou reservar assentos.
 - Recebe e exibe as respostas do servidor.

Instruções de Uso

Execução

Abra o terminal e inicie o servidor:

```
python servidor.py
```

1.

Em outro terminal (ou máquina na rede), inicie o cliente:

```
python cliente.py
```

2.

3. **No cliente, siga as instruções:**

- Digite `ver` para visualizar os assentos disponíveis.

Digite os números dos assentos separados por vírgula para reservar, por exemplo:

10,15,22

-
- Digite `sair` para encerrar a conexão.

Exemplo de uso:

Digite 'ver' para ver os assentos,
ou números dos assentos separados por vírgula para reservar (ex: 10,12),
ou 'sair' para sair:

> ver

Assento 1: Livre

Assento 2: Reservado

...

> 3,4

✅ Reserva confirmada.

Sincronização

O seguinte trecho garante segurança na reserva:

with lock:

verificação + marcação dos assentos

Isso impede que duas threads alterem o mesmo recurso simultaneamente, evitando todos os problemas descritos no enunciado (condições de corrida, inconsistência e perda de dados).

