



**INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA BAHIA
CAMPUS - SANTO ANTÔNIO DE JESUS - BAHIA**

**CURSO DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE
SISITEMAS**

MARCELO DE JESUS

RONALDO CORREIA

**ATIVIDADE DE SISTEMAS DISTRIBUIDO- PROJETO FINAL –
PLATAFORMA DISTRIBUÍDA DE PROCESSAMENTO
COLABORATIVO DE TAREFAS**

SANTO ANTONIO DE JESUS – BA 2025

MARCELO DE JESUS

RONALDO CORREIA

**ATIVIDADE DE SISTEMAS DISTRIBUIDO- PROJETO FINAL –
PLATAFORMA DISTRIBUÍDA DE PROCESSAMENTO
COLABORATIVO DE TAREFAS**

Relatório técnico da
atividade final de Sistemas
Distribuído- Projeto final –
Plataforma Distribuída de
Processamento Colaborativo de
Tarefas, elaborado como
requisito parcial de avaliação
para a disciplina de Padrões
de Projeto, ministrada pelo Prof.
Felipe Silva.

SANTO ANTONIO DE JESUS – BA

2025

1. INTRODUÇÃO

Os sistemas distribuídos constituem um dos pilares da computação moderna, permitindo que múltiplos recursos computacionais trabalhem de forma cooperativa para alcançar maior desempenho, escalabilidade, tolerância a falhas e disponibilidade. Em contraste aos sistemas centralizados, nos quais a queda de um único componente pode comprometer toda a operação, a distribuição das responsabilidades entre diferentes processos e máquinas possibilita maior robustez e flexibilidade no tratamento de tarefas críticas.

Neste contexto, a presente atividade acadêmica teve como objetivo o desenvolvimento de uma aplicação prática que simula um ambiente distribuído, no qual clientes interagem com um orquestrador principal responsável por gerenciar tarefas e coordenar sua execução entre múltiplos *workers*. A arquitetura projetada incorpora ainda mecanismos de replicação de estado e um orquestrador de backup, capaz de assumir automaticamente em caso de falha do servidor principal (*failover*), assegurando assim a continuidade dos serviços.

Para reforçar a confiabilidade do sistema, foram implementados protocolos de monitoramento baseados em *heartbeat*, que permitem identificar a disponibilidade dos *workers* e detectar falhas de execução. Além disso, utilizou-se um algoritmo de balanceamento de carga no estilo *Round Robin*, visando distribuir as tarefas de forma justa e eficiente entre os nós disponíveis. A comunicação entre os componentes foi realizada por meio de *sockets* e mensagens serializadas em formato JSON, garantindo interoperabilidade e simplicidade na troca de dados.

Com esta implementação, busca-se não apenas demonstrar conceitos teóricos de sistemas distribuídos, mas também proporcionar uma aplicação prática que evidencia a importância de mecanismos de tolerância a falhas, autenticação de usuários e coordenação descentralizada. O trabalho, portanto, contribui para a consolidação do aprendizado acerca de arquiteturas resilientes e escaláveis, aplicáveis tanto em ambientes acadêmicos quanto em cenários reais de processamento distribuído.

2. FUNDAMENTAÇÃO TEORICA

Sistemas distribuídos são compostos por múltiplos processos independentes que se comunicam e cooperam para executar tarefas de forma coordenada. Ao contrário de sistemas centralizados, onde um único servidor realiza todas as operações, a distribuição permite escalabilidade, maior disponibilidade e tolerância a falhas, características essenciais em aplicações modernas de computação em nuvem, redes de sensores, processamento paralelo e plataformas colaborativas.

2.1 Orquestração de Tarefas

A orquestração de tarefas é um mecanismo pelo qual um componente central (orquestrador) gerencia a execução de trabalhos distribuídos entre diversos nós de processamento (workers). Esse conceito é fundamental para garantir que as tarefas sejam distribuídas eficientemente, que os estados sejam monitorados em tempo real e que falhas sejam detectadas e corrigidas rapidamente. Em sistemas críticos, o uso de um **orquestrador secundário** (backup) permite o *failover* automático, assegurando a continuidade do processamento mesmo diante da indisponibilidade do orquestrador principal.

2.2 Balanceamento de Carga

O balanceamento de carga é a estratégia utilizada para distribuir tarefas ou requisições de forma eficiente entre múltiplos nós, evitando sobrecarga em um único ponto e maximizando o desempenho do sistema. Entre as políticas de balanceamento, destaca-se:

- **Round Robin:** tarefas são atribuídas ciclicamente aos workers, garantindo distribuição uniforme.
- **Least Load:** tarefas são direcionadas para o worker com menor carga atual, minimizando tempo de execução.
- **Aleatória:** tarefas são distribuídas de forma probabilística, simplificando a implementação.

Neste projeto, adotou-se a política *Round Robin*, por sua simplicidade, previsibilidade e facilidade de implementação em cenários acadêmicos.

2.3 Tolerância a Falhas e Failover

A tolerância a falhas é um requisito essencial em sistemas distribuídos. O monitoramento de *heartbeat* permite detectar indisponibilidade de nós e orquestradores, enquanto a replicação de estado assegura que dados críticos não sejam perdidos. Quando um worker falha, suas tarefas são redistribuídas automaticamente, garantindo continuidade do processamento. De forma similar, se o orquestrador principal se torna indisponível, o backup assume suas funções de maneira transparente, mantendo o sistema operacional.

2.4 Relógios Lógicos de Lamport

Em sistemas distribuídos, eventos podem ocorrer de forma simultânea e independente, dificultando a ordenação temporal. Os **relógios lógicos de Lamport** são algoritmos que atribuem timestamps a eventos, permitindo estabelecer uma ordem parcial consistente entre eles. Esta ordenação é essencial para manter o estado global consistente, detectar conflitos e sincronizar ações entre orquestradores e workers.

2.5 Autenticação e Segurança Básica

Para controlar o acesso às tarefas e preservar a integridade do sistema, implementou-se uma autenticação básica de clientes baseada em usuário e senha. Cada cliente autenticado recebe um token único que valida sua comunicação com o orquestrador. Esse mecanismo simples demonstra conceitos de segurança em sistemas distribuídos e assegura que apenas usuários autorizados possam submeter tarefas ou consultar status.

2.6 Comunicação entre Processos

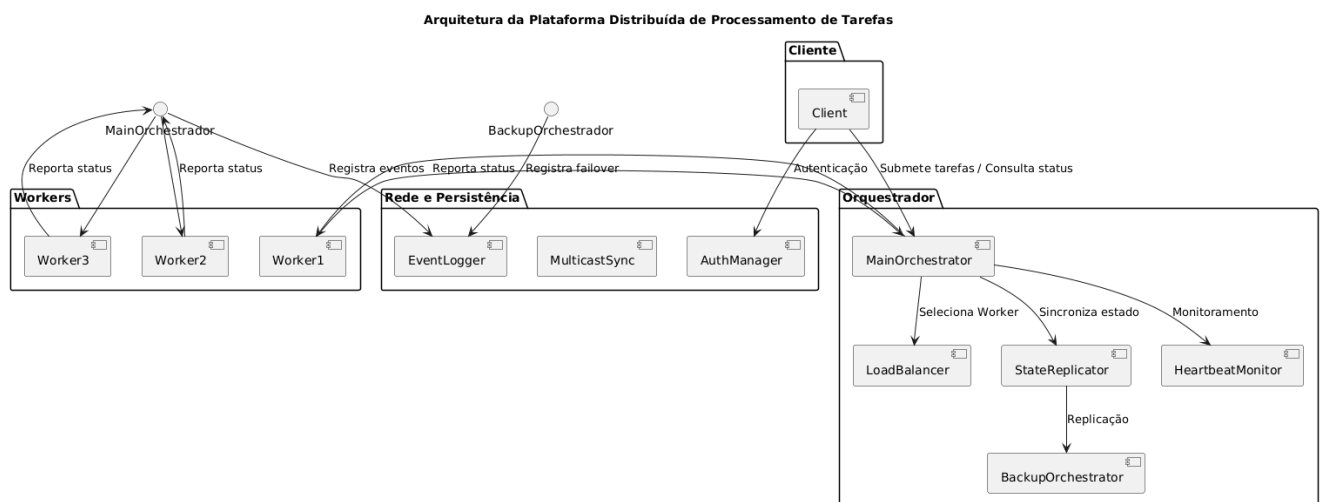
A comunicação entre os componentes do sistema foi implementada utilizando **sockets TCP** para mensagens confiáveis entre clientes, orquestradores e workers, e **UDP Multicast** para sincronização de estado entre orquestradores. A serialização de objetos em **JSON** permite interoperabilidade e simplicidade na troca de informações, facilitando a manutenção e expansão do sistema.

3. ARQUITETURA DO SISTEMA

A plataforma distribuída de processamento colaborativo de tarefas foi projetada para simular um sistema real de execução distribuída, integrando conceitos de escalabilidade,

tolerância a falhas e monitoramento em tempo real. A arquitetura do sistema foi dividida em quatro componentes principais: **Clientes**, **Orquestrador Principal**, **Orquestrador Backup** e **Workers**, cada um encapsulado em pacotes específicos para modularidade e manutenção.

3.1 Diagrama de componentes- Arquitetura Geral

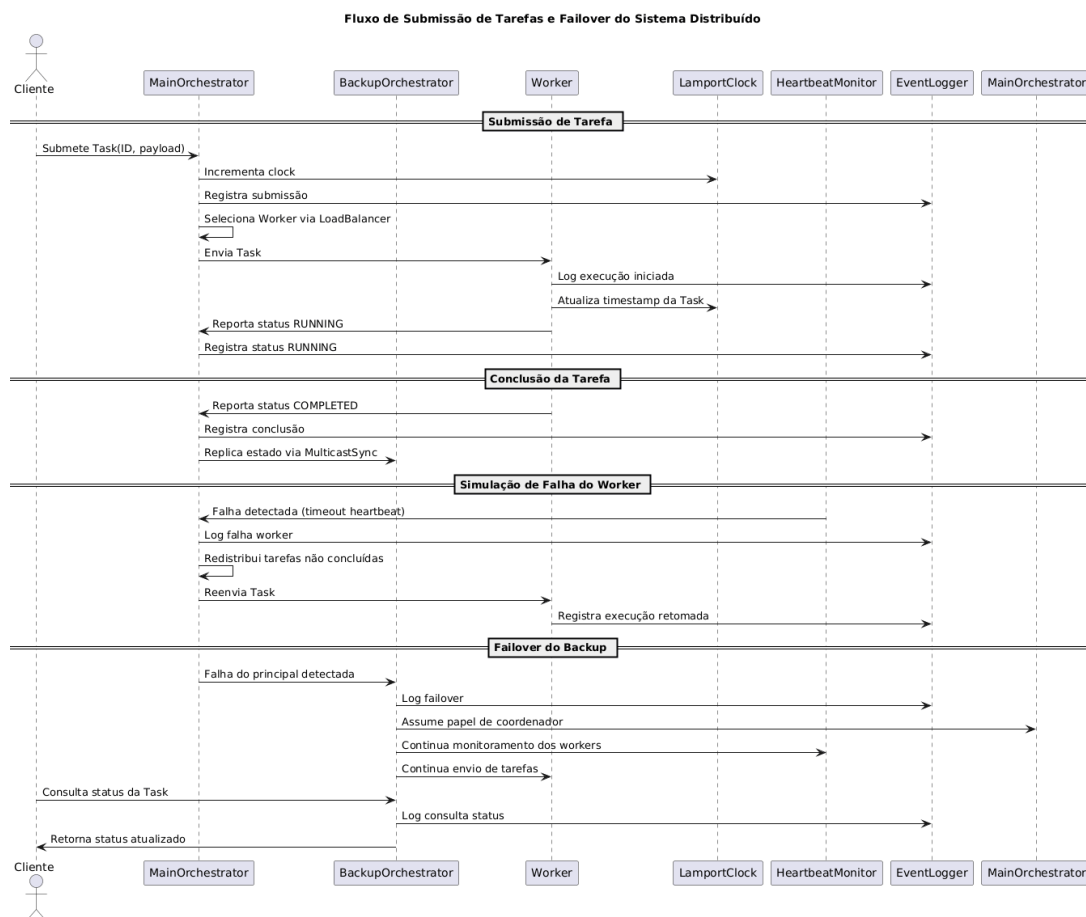


Explicação do Diagrama

1. **Cliente:** envia tarefas e consulta status, autenticando-se via **AuthManager**.
2. **Orquestrador Principal (MainOrchestrator):** distribui tarefas, usa **LoadBalancer**, replica estado, monitora falhas (**HeartbeatMonitor**) e registra eventos (**EventLogger**).
3. **BackupOrchestrator:** mantém cópia do estado global via **StateReplicator** e assume o papel do principal em caso de falha.
4. **Workers:** executam tarefas e reportam status, registrando logs.

5. **Componentes Comuns:** Task, LamportClock e HeartbeatMonitor são usados por todos os módulos.
6. **Rede e Autenticação:** AuthManager para login/token e MulticastSync para sincronização do estado global.

3.2 Diagrama de Sequência

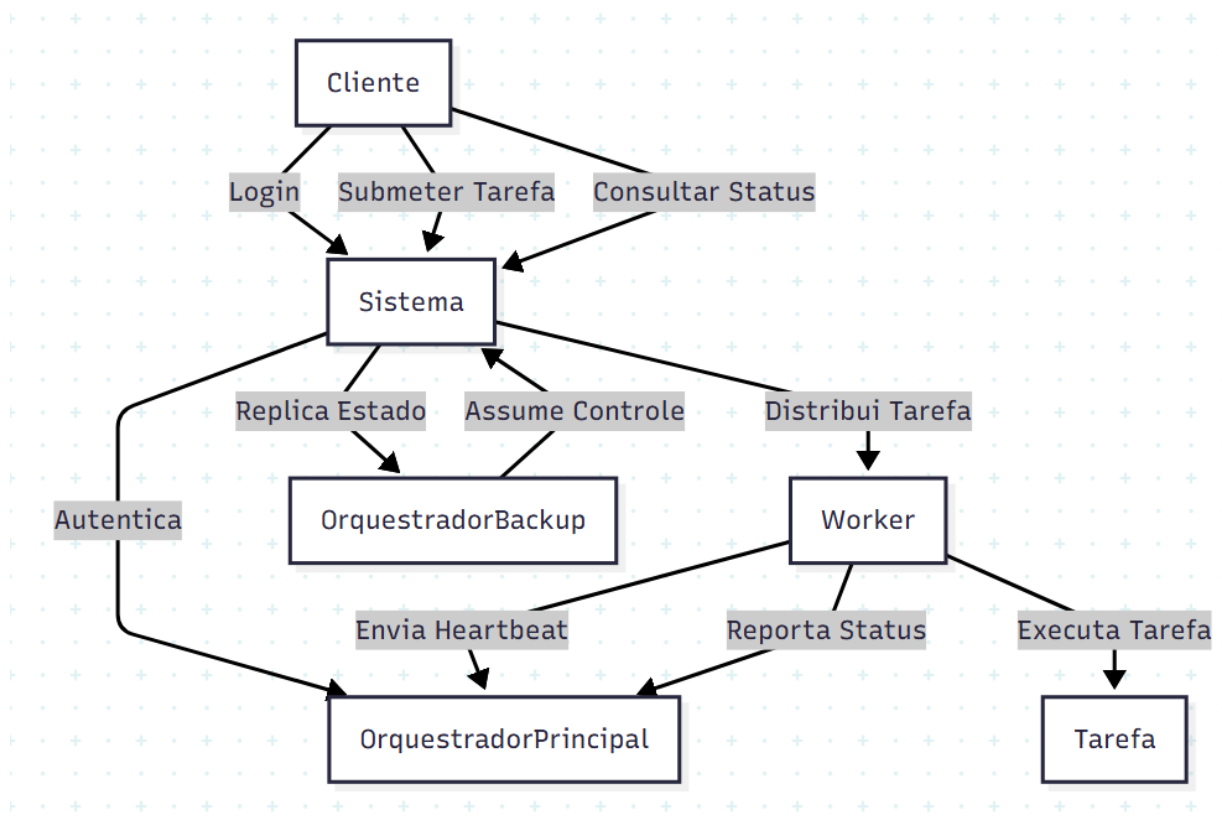


Explicação:

1. **Submissão de tarefa:** Cliente envia a tarefa autenticada → Orquestrador incrementa relógio lógico → registra evento → seleciona worker → envia tarefa.

2. **Execução do worker:** Worker executa a tarefa, atualiza status via Lamport → reporta status RUNNING e depois COMPLETED → logs centralizados.
3. **Falha do worker:** Heartbeat detecta falha → Orquestrador registra e redistribui tarefas → reenvio para outro worker.
4. **Failover do backup:** Backup detecta falha do principal → assume orquestrador → continua distribuição e monitoramento.
5. **Consulta de status:** Cliente pode consultar status atualizado, mesmo após failover, garantindo consistência.

3.3 Diagrama de casos de uso



Explicação:

1. Cliente

- 1.1 Realiza o **Login** no sistema.
- 1.2 **Submete tarefas** para processamento.

- 1.3 **Consulta o status** das tarefas enviadas.

2. Sistema (Orquestrador Ativo)

- 2.1 Recebe as requisições do cliente.
- 2.2 **Autentica** o usuário via AuthManager.
- 2.3 **Distribui tarefas** para os workers usando o LoadBalancer.
- 2.4 **Monitora os workers** através do HeartbeatMonitor.
- 2.5 **Atualiza o status** das tarefas com base nas respostas dos workers.
- 2.6 **Replica o estado** atual para o BackupOrchestrator via StateReplicator.

3. OrquestradorPrincipal

- 3.1 Autentica o cliente.
- 3.2 Recebe e distribui tarefas.
- 3.3 Envia **heartbeat** para o BackupOrchestrator para indicar que está ativo.
- 3.4 Sincroniza o estado das tarefas com o backup.

4. OrquestradorBackup

- 4.1 Recebe replicações do estado do principal.
- 4.2 Monitora o heartbeat do principal.
- 4.3 **Assume o controle** automaticamente se o principal falhar.
- 4.4 Redistribui tarefas pendentes ou falhadas.

5. Worker

- 5.1 Recebe tarefas do orquestrador.
- 5.2 **Executa a tarefa** e atualiza seu status (RUNNING, COMPLETED, FAILED).
- 5.3 Envia **heartbeat** periódico ao orquestrador.
- 5.4 **Reporta o status** da tarefa após execução.

6. Tarefa

- 6.1 Criada pelo cliente com ID e conteúdo.
- 6.2 Atribuída a um worker pelo orquestrador.

- 6.3 Passa por estados: PENDING → RUNNING → COMPLETED ou FAILED.
- 6.4 É monitorada e registrada no EventLogger

3.4 Estrutura Modular do Sistema

O código-fonte foi organizado em pacotes funcionais:

- **client:** Contém a classe Client, responsável por fornecer a interface de linha de comando aos usuários, permitindo autenticação, submissão de tarefas e consulta de status em tempo real.
- **common:** Inclui classes compartilhadas entre componentes, como Task (representação de tarefas), HeartbeatMonitor (monitoramento de falhas dos workers) e LamportClock (relógio lógico de Lamport para ordenação de eventos).
- **logs:** Contém EventLogger, utilizado para registrar todos os eventos relevantes, como submissão, execução, falha e failover, garantindo rastreabilidade e evidência de execução.
- **network:** Abriga AuthManager para autenticação de usuários e MulticastSync para sincronização do estado global entre orquestradores via UDP multicast.
- **orchestrator:** Compreende a lógica principal de distribuição de tarefas (MainOrchestrator), balanceamento de carga (LoadBalancer), replicação de estado (StateReplicator) e controle de failover (BackupOrchestrator).
- **worker:** Contém Worker e WorkerNode, responsáveis pela execução das tarefas atribuídas, envio de heartbeat e reporte de status ao orquestrador.

3.5 Diagramas de Componentes

A arquitetura modular permite que cada componente funcione de forma independente, comunicando-se por canais bem definidos. A comunicação cliente-orquestrador é feita via **sockets TCP**, garantindo confiabilidade na transmissão de tarefas e consultas. A sincronização entre orquestradores utiliza **UDP multicast**, permitindo replicação de estado de forma leve e eficiente.

Justificativa:

- A separação por pacotes aumenta a manutenibilidade do código e permite futura escalabilidade (adição de mais workers ou orquestradores).
- O uso de JSON para serialização de objetos facilita interoperabilidade e teste de integração entre módulos.
- O monitoramento por heartbeat e replicação de estado garante **tolerância a falhas**, cumprindo os requisitos do projeto.

3.6 Políticas e Algoritmos

- **Balanceamento de Carga:** Implementado com a política **Round Robin**, que distribui tarefas de forma cíclica e previsível entre os workers, evitando sobrecarga em um único nó.
- **Tolerância a Falhas:** O `HeartbeatMonitor` detecta falhas em workers e orquestradores. Tarefas em execução em workers inativos são redistribuídas automaticamente, enquanto o backup assume o papel de orquestrador principal em caso de falha, mantendo o estado global consistente.
- **Relógios Lógicos de Lamport:** Garantem que eventos distribuídos, como submissão, execução, falha e recuperação de tarefas, sejam ordenados de forma consistente, evitando conflitos e inconsistências no estado global.

3.7 Logs e Monitoramento

O sistema registra eventos importantes por meio do `EventLogger`, incluindo:

- Submissão de tarefas pelo cliente.
- Distribuição das tarefas pelo orquestrador.
- Conclusão ou falha em workers.
- Failover do backup assumindo o papel de principal.
- Redistribuição de tarefas não concluídas.

Justificativa:

- Os logs permitem auditoria do sistema, verificação de funcionamento e geração de evidências para avaliação acadêmica.

- Auxiliam na análise de performance, identificação de gargalos e validação da consistência do estado global.

3.8 Considerações sobre a Arquitetura

A escolha de uma arquitetura distribuída modular com componentes independentes foi motivada por:

1. **Escalabilidade:** Facilita a adição de novos workers sem alterar a lógica do orquestrador principal.
2. **Resiliência:** O sistema mantém-se funcional mesmo diante de falhas de nós ou do coordenador principal.
3. **Manutenibilidade:** Pacotes separados permitem atualização de módulos específicos sem impactar outros componentes.
4. **Transparência:** Clientes não precisam conhecer detalhes internos de distribuição; apenas interação via interface de linha de comando autenticada.

4. ESCOLHA DE PROTOCOLOS, ALGORITMOS E POLÍTICAS DE BALANCEAMENTO

Para o desenvolvimento da plataforma distribuída, a seleção de protocolos de comunicação, algoritmos de orquestração e políticas de balanceamento foi realizada com base em critérios de confiabilidade, desempenho e consistência de estado.

- **Protocolos de Comunicação:**
A interação entre clientes e o orquestrador principal foi implementada utilizando **TCP (Transmission Control Protocol)**. A escolha do TCP justifica-se pela necessidade de garantir entrega confiável das mensagens, controle de fluxo e correção de erros, fatores

essenciais para manter a integridade das tarefas submetidas pelos clientes. Já a comunicação entre o orquestrador principal e o backup utiliza **UDP Multicast**, permitindo a replicação simultânea do estado global para múltiplos orquestradores de forma eficiente. Embora o UDP não forneça garantias de entrega, sua utilização neste contexto é justificada pela frequência contínua de sincronização e pela tolerância a perdas ocasionais, características compatíveis com o modelo de failover planejado.

- **Algoritmos:**

O sistema adota o algoritmo **Round Robin** como política de balanceamento de carga, distribuindo tarefas sequencialmente entre os workers disponíveis. Esta abordagem simples e determinística garante uma distribuição equitativa, reduzindo a possibilidade de sobrecarga em um único nó e facilitando a previsibilidade do comportamento do sistema. Além disso, a plataforma utiliza **heartbeat monitor** para detecção de falhas nos workers e algoritmos de **redistribuição de tarefas**, assegurando tolerância a falhas e continuidade de processamento. Para manter a consistência temporal das operações distribuídas, empregou-se o **relógio lógico de Lamport**, permitindo ordenar eventos entre os diferentes nós e evitando inconsistências na execução de tarefas.

- **Políticas de Balanceamento:**

O Round Robin foi escolhido por sua simplicidade e eficiência em cenários de cargas relativamente homogêneas. Apesar de não considerar o estado individual de cada worker, ele proporciona uma distribuição uniforme e previsível, adequada ao escopo acadêmico do projeto. Em trabalhos futuros, políticas mais sofisticadas, como **Least Load** ou balanceamento adaptativo baseado em métricas de desempenho, poderiam ser adotadas para otimização em ambientes com cargas variáveis.

5. EVIDÊNCIAS E EXECUÇÃO

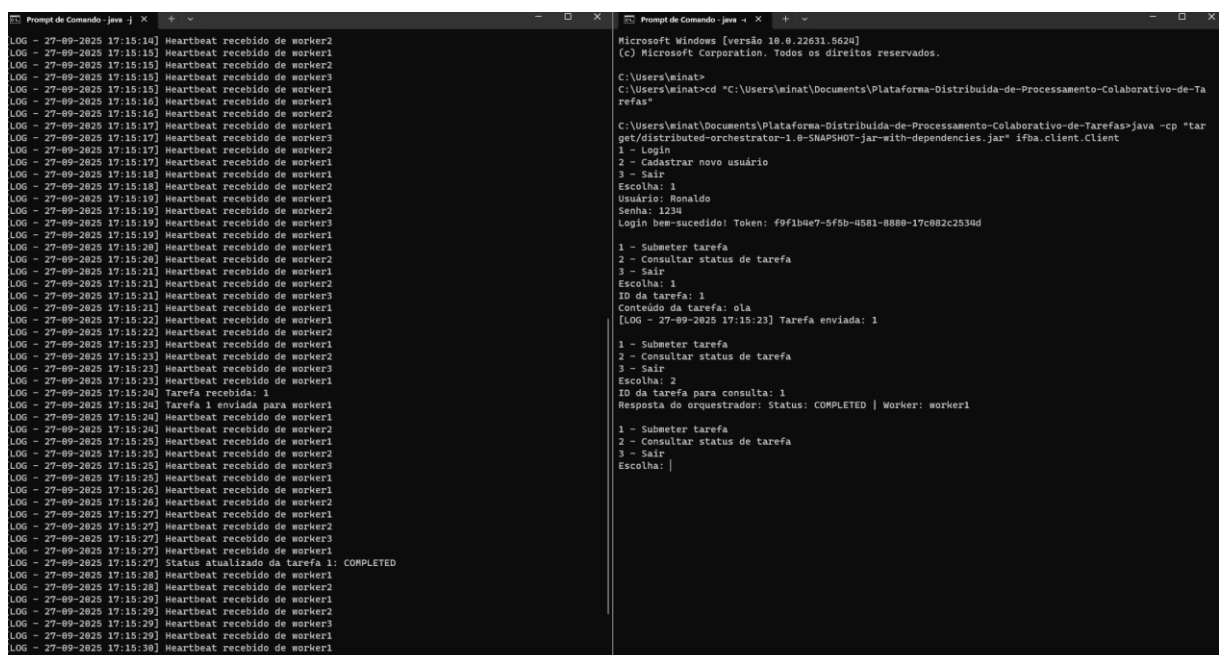
A validação do sistema distribuído foi realizada por meio de testes práticos que simulam cenários reais de processamento colaborativo. Para tanto, foi utilizado um conjunto de **três workers**, um **orquestrador principal** e um **orquestrador de backup**, permitindo verificar a correta distribuição de tarefas, o monitoramento do estado global e a tolerância a falhas.

Durante os testes, os seguintes procedimentos foram observados:

1. **Submissão de Tarefas:** Clientes autenticados enviaram tarefas para o orquestrador principal, que realizou a distribuição automática aos workers de acordo com a política de balanceamento Round Robin.
2. **Monitoramento de Status:** O sistema forneceu feedback em tempo real sobre o estado das tarefas, permitindo a verificação de execução, conclusão ou falhas de forma transparente.
3. **Falhas Simuladas:** Falhas de workers foram simuladas interrompendo temporariamente os heartbeats. O orquestrador principal detectou estas falhas e redistribuiu as tarefas não concluídas para outros workers ativos.
4. **Failover:** A falha do orquestrador principal foi simulada, resultando na ativação automática do orquestrador backup, que assumiu a coordenação das tarefas sem perda de estado global.

Os **logs centralizados** gerados durante a execução registraram todos os eventos relevantes, incluindo submissão de tarefas, distribuição, execução, falhas e redistribuição, todos ordenados por **timestamps do relógio lógico de Lamport**, garantindo a rastreabilidade e consistência do sistema. Esses registros foram documentados como evidências em anexos do relatório e servem como comprovação do funcionamento correto da plataforma.

5.1 Prints e Execução



The image displays two side-by-side Windows command prompt windows. The left window shows a continuous stream of log messages from a Java application, indicating heartbeat reception from three workers (worker1, worker2, worker3) at regular intervals. The right window shows a user interacting with the application via a command-line interface. The user enters commands to login, submit a task, and check its status. The application responds with a token, task ID, and status (COMPLETED), along with worker information.

```
Prompt de Comando - java - j x
LOG - 27-09-2025 17:15:14] Heartbeat recebido de worker2
LOG - 27-09-2025 17:15:15] Heartbeat recebido de worker1
LOG - 27-09-2025 17:15:15] Heartbeat recebido de worker2
LOG - 27-09-2025 17:15:15] Heartbeat recebido de worker3
LOG - 27-09-2025 17:15:15] Heartbeat recebido de worker1
LOG - 27-09-2025 17:15:16] Heartbeat recebido de worker1
LOG - 27-09-2025 17:15:16] Heartbeat recebido de worker2
LOG - 27-09-2025 17:15:16] Heartbeat recebido de worker2
LOG - 27-09-2025 17:15:17] Heartbeat recebido de worker1
LOG - 27-09-2025 17:15:17] Heartbeat recebido de worker3
LOG - 27-09-2025 17:15:17] Heartbeat recebido de worker2
LOG - 27-09-2025 17:15:17] Heartbeat recebido de worker1
LOG - 27-09-2025 17:15:18] Heartbeat recebido de worker1
LOG - 27-09-2025 17:15:18] Heartbeat recebido de worker2
LOG - 27-09-2025 17:15:19] Heartbeat recebido de worker1
LOG - 27-09-2025 17:15:19] Heartbeat recebido de worker2
LOG - 27-09-2025 17:15:19] Heartbeat recebido de worker3
LOG - 27-09-2025 17:15:19] Heartbeat recebido de worker1
LOG - 27-09-2025 17:15:20] Heartbeat recebido de worker1
LOG - 27-09-2025 17:15:20] Heartbeat recebido de worker2
LOG - 27-09-2025 17:15:21] Heartbeat recebido de worker1
LOG - 27-09-2025 17:15:21] Heartbeat recebido de worker3
LOG - 27-09-2025 17:15:21] Heartbeat recebido de worker1
LOG - 27-09-2025 17:15:22] Heartbeat recebido de worker1
LOG - 27-09-2025 17:15:22] Heartbeat recebido de worker2
LOG - 27-09-2025 17:15:22] Heartbeat recebido de worker1
LOG - 27-09-2025 17:15:23] Heartbeat recebido de worker1
LOG - 27-09-2025 17:15:23] Heartbeat recebido de worker2
LOG - 27-09-2025 17:15:23] Heartbeat recebido de worker3
LOG - 27-09-2025 17:15:23] Heartbeat recebido de worker1
LOG - 27-09-2025 17:15:24] Tarefa recebida: 1
LOG - 27-09-2025 17:15:24] Tarefa 1 enviada para worker1
LOG - 27-09-2025 17:15:24] Heartbeat recebido de worker1
LOG - 27-09-2025 17:15:24] Heartbeat recebido de worker2
LOG - 27-09-2025 17:15:25] Heartbeat recebido de worker1
LOG - 27-09-2025 17:15:25] Heartbeat recebido de worker2
LOG - 27-09-2025 17:15:25] Heartbeat recebido de worker3
LOG - 27-09-2025 17:15:25] Heartbeat recebido de worker1
LOG - 27-09-2025 17:15:26] Heartbeat recebido de worker1
LOG - 27-09-2025 17:15:26] Heartbeat recebido de worker2
LOG - 27-09-2025 17:15:26] Heartbeat recebido de worker2
LOG - 27-09-2025 17:15:27] Heartbeat recebido de worker2
LOG - 27-09-2025 17:15:27] Heartbeat recebido de worker2
LOG - 27-09-2025 17:15:27] Heartbeat recebido de worker3
LOG - 27-09-2025 17:15:27] Heartbeat recebido de worker1
LOG - 27-09-2025 17:15:27] Status atualizado da tarefa 1: COMPLETED
LOG - 27-09-2025 17:15:28] Heartbeat recebido de worker1
LOG - 27-09-2025 17:15:28] Heartbeat recebido de worker2
LOG - 27-09-2025 17:15:29] Heartbeat recebido de worker1
LOG - 27-09-2025 17:15:29] Heartbeat recebido de worker2
LOG - 27-09-2025 17:15:29] Heartbeat recebido de worker3
LOG - 27-09-2025 17:15:29] Heartbeat recebido de worker1
LOG - 27-09-2025 17:15:30] Heartbeat recebido de worker1
```

```
Prompt de Comando - java - j x
Microsoft Windows [versão 10.0.22631.5624]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\winat>
C:\Users\winat>cd "C:\Users\winat\Documents\Plataforma-Distribuida-de-Processamento-Colaborativo-de-Ta
refas"

C:\Users\winat\Documents\Plataforma-Distribuida-de-Processamento-Colaborativo-de-Tarefas>java -cp "tar
get/distributed-orchestrator-1.0-SNAPSHOT-jar-with-dependencies.jar" ifba.client.Client

1 - Login
2 - Cadastrar novo usuário
3 - Sair
Escolha: 1
Usuário: Ronaldo
Senha: 1234
Login bem-sucedido! Token: f9f1b4e7-5f5b-4581-8888-17c882c2534d

1 - Submeter tarefa
2 - Consultar status de tarefa
3 - Sair
Escolha: 1
ID da tarefa: 1
Conteúdo da tarefa: ola
[LOG - 27-09-2025 17:15:23] Tarefa enviada: 1

1 - Submeter tarefa
2 - Consultar status de tarefa
3 - Sair
Escolha: 2
ID da tarefa para consulta: 1
Resposta do orquestrador: Status: COMPLETED | Worker: worker1

1 - Submeter tarefa
2 - Consultar status de tarefa
3 - Sair
Escolha: |
```

| | |
|---|--|
| <pre>Prompt de Comando - java - X + - Microsoft Windows [versão 10.0.22631.5624] (c) Microsoft Corporation. Todos os direitos reservados. C:\Users\minat>cd "C:\Users\minat\Documents\Plataforma-Distribuida-de-Processamento-Colaborativo-de-Tarefas" C:\Users\minat\Documents\Plataforma-Distribuida-de-Processamento-Colaborativo-de-Tarefas>java -cp "target\distributed-orchestrator-1.0-SNAPSHOT-jar-with-dependencies.jar" ifba.BackupMain Backup orchestrator está escutando na porta 6080... [LOG - 27-09-2025 17:15:02] Orquestrador escutando na porta 6080</pre> | <pre>Prompt de Comando - java - X + - Microsoft Windows [versão 10.0.22631.5624] (c) Microsoft Corporation. Todos os direitos reservados. C:\Users\minat>cd "C:\Users\minat\Documents\Plataforma-Distribuida-de-Processamento-Colaborativo-de-Tarefas" C:\Users\minat\Documents\Plataforma-Distribuida-de-Processamento-Colaborativo-de-Tarefas>java -cp "target\distributed-orchestrator-1.0-SNAPSHOT-jar-with-dependencies.jar" ifba.WorkerMain worker1 6081 [LOG - 27-09-2025 17:15:03] Worker worker1 aguardando tarefas na porta 6081 [LOG - 27-09-2025 17:15:24] Worker worker1 iniciou tarefa 1 [LOG - 27-09-2025 17:15:27] Worker worker1 concluiu tarefa 1</pre> |
| <pre>Prompt de Comando - java - X + - Microsoft Windows [versão 10.0.22631.5624] (c) Microsoft Corporation. Todos os direitos reservados. C:\Users\minat>cd "C:\Users\minat\Documents\Plataforma-Distribuida-de-Processamento-Colaborativo-de-Tarefas" C:\Users\minat\Documents\Plataforma-Distribuida-de-Processamento-Colaborativo-de-Tarefas>java -cp "target\distributed-orchestrator-1.0-SNAPSHOT-jar-with-dependencies.jar" ifba.WorkerMain worker2 6082 [LOG - 27-09-2025 17:15:04] Erro no worker worker1: Address already in use: bind [LOG - 27-09-2025 17:15:04] Worker worker2 aguardando tarefas na porta 6082</pre> | <pre>Prompt de Comando - java - X + - Microsoft Windows [versão 10.0.22631.5624] (c) Microsoft Corporation. Todos os direitos reservados. C:\Users\minat>cd "C:\Users\minat\Documents\Plataforma-Distribuida-de-Processamento-Colaborativo-de-Tarefas" C:\Users\minat\Documents\Plataforma-Distribuida-de-Processamento-Colaborativo-de-Tarefas>java -cp "target\distributed-orchestrator-1.0-SNAPSHOT-jar-with-dependencies.jar" ifba.WorkerMain worker3 6083 [LOG - 27-09-2025 17:15:05] Erro no worker worker1: Address already in use: bind [LOG - 27-09-2025 17:15:05] Erro no worker worker2: Address already in use: bind [LOG - 27-09-2025 17:15:05] Worker worker3 aguardando tarefas na porta 6083</pre> |

Microsoft Windows [versão 10.0.22631.5624]

(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\minat>cd "C:\Users\minat\Documents\Plataforma-Distribuida-de-Processamento-Colaborativo-de-Tarefas"

C:\Users\minat\Documents\Plataforma-Distribuida-de-Processamento-Colaborativo-de-Tarefas>java -jar target/distributed-orchestrator-1.0-SNAPSHOT-jar-with-dependencies.jar

Comandos:

- 1 - Enviar tarefa
- 2 - Simular falha de worker
- 3 - Ver status das tarefas
- 4 - Sair

Escolha: [LOG - 27-09-2025 17:14:27] Orquestrador escutando na porta 5000

[LOG - 27-09-2025 17:15:03] Heartbeat recebido de worker1

[LOG - 27-09-2025 17:15:04] Heartbeat recebido de worker1

[LOG - 27-09-2025 17:15:04] Heartbeat recebido de worker2

[LOG - 27-09-2025 17:15:05] Heartbeat recebido de worker1

[LOG - 27-09-2025 17:15:05] Heartbeat recebido de worker2

[LOG - 27-09-2025 17:15:05] Heartbeat recebido de worker3

[LOG - 27-09-2025 17:15:05] Heartbeat recebido de worker1

[LOG - 27-09-2025 17:15:06] Heartbeat recebido de worker1

[LOG - 27-09-2025 17:15:06] Heartbeat recebido de worker2

[LOG - 27-09-2025 17:15:07] Heartbeat recebido de worker1

[LOG - 27-09-2025 17:15:07] Heartbeat recebido de worker3

[LOG - 27-09-2025 17:15:07] Heartbeat recebido de worker2

[LOG - 27-09-2025 17:15:07] Heartbeat recebido de worker1

[LOG - 27-09-2025 17:15:08] Heartbeat recebido de worker1

[LOG - 27-09-2025 17:15:08] Heartbeat recebido de worker2

[LOG - 27-09-2025 17:15:09] Heartbeat recebido de worker1

[LOG - 27-09-2025 17:15:09] Heartbeat recebido de worker2

[LOG - 27-09-2025 17:15:09] Heartbeat recebido de worker3

[LOG - 27-09-2025 17:15:09] Heartbeat recebido de worker1

[LOG - 27-09-2025 17:15:10] Heartbeat recebido de worker1

[LOG - 27-09-2025 17:15:10] Heartbeat recebido de worker2

[LOG - 27-09-2025 17:15:11] Heartbeat recebido de worker1

[LOG - 27-09-2025 17:15:11] Heartbeat recebido de worker2

[LOG - 27-09-2025 17:15:11] Heartbeat recebido de worker3

[LOG - 27-09-2025 17:15:11] Heartbeat recebido de worker1

[LOG - 27-09-2025 17:15:12] Heartbeat recebido de worker1

[LOG - 27-09-2025 17:15:12] Heartbeat recebido de worker2

[LOG - 27-09-2025 17:15:13] Heartbeat recebido de worker1

[LOG - 27-09-2025 17:15:13] Heartbeat recebido de worker2

[LOG - 27-09-2025 17:15:13] Heartbeat recebido de worker3

[LOG - 27-09-2025 17:15:13] Heartbeat recebido de worker1

[LOG - 27-09-2025 17:15:14] Heartbeat recebido de worker1

[LOG - 27-09-2025 17:15:14] Heartbeat recebido de worker2

[LOG - 27-09-2025 17:15:15] Heartbeat recebido de worker1

[LOG - 27-09-2025 17:15:15] Heartbeat recebido de worker2

[LOG - 27-09-2025 17:15:15] Heartbeat recebido de worker3


```
[LOG - 27-09-2025 17:16:37] Heartbeat recebido de worker1
2
Qual worker deseja simular falha? (worker1, worker2, worker3): [LOG - 27-09-2025 17:16:38] Heartbeat
recebido de worker1
[LOG - 27-09-2025 17:16:38] Heartbeat recebido de worker2
[LOG - 27-09-2025 17:16:39] Heartbeat recebido de worker1
[LOG - 27-09-2025 17:16:39] Heartbeat recebido de worker3
[LOG - 27-09-2025 17:16:39] Heartbeat recebido de worker2
[LOG - 27-09-2025 17:16:39] Heartbeat recebido de worker1
[LOG - 27-09-2025 17:16:40] Heartbeat recebido de worker1
[LOG - 27-09-2025 17:16:40] Heartbeat recebido de worker2
w[LOG - 27-09-2025 17:16:41] Heartbeat recebido de worker1
o[LOG - 27-09-2025 17:16:41] Heartbeat recebido de worker3
[LOG - 27-09-2025 17:16:41] Heartbeat recebido de worker2
[LOG - 27-09-2025 17:16:41] Heartbeat recebido de worker1
rker[LOG - 27-09-2025 17:16:42] Heartbeat recebido de worker1
[LOG - 27-09-2025 17:16:42] Heartbeat recebido de worker2
1
Simulando falha: heartbeat de worker1 interrompido por 10 segundos...

Comandos:
1 - Enviar tarefa
2 - Simular falha de worker
3 - Ver status das tarefas
4 - Sair
Escolha: [LOG - 27-09-2025 17:16:43] Heartbeat recebido de worker1
[LOG - 27-09-2025 17:16:43] Heartbeat recebido de worker3
[LOG - 27-09-2025 17:16:43] Heartbeat recebido de worker2
[LOG - 27-09-2025 17:16:43] Heartbeat recebido de worker1
[LOG - 27-09-2025 17:16:44] Heartbeat recebido de worker1
[LOG - 27-09-2025 17:16:44] Heartbeat recebido de worker2
[LOG - 27-09-2025 17:16:45] Heartbeat recebido de worker1
[LOG - 27-09-2025 17:16:45] Heartbeat recebido de worker3
[LOG - 27-09-2025 17:16:45] Heartbeat recebido de worker2
[LOG - 27-09-2025 17:16:45] Heartbeat recebido de worker1
[LOG - 27-09-2025 17:16:46] Heartbeat recebido de worker1
[LOG - 27-09-2025 17:16:46] Heartbeat recebido de worker2
[LOG - 27-09-2025 17:16:47] Heartbeat recebido de worker1
[LOG - 27-09-2025 17:16:47] Heartbeat recebido de worker3
[LOG - 27-09-2025 17:16:47] Heartbeat recebido de worker2
[LOG - 27-09-2025 17:16:47] Heartbeat recebido de worker1
[LOG - 27-09-2025 17:16:48] Heartbeat recebido de worker1
[LOG - 27-09-2025 17:16:48] Heartbeat recebido de worker2
[LOG - 27-09-2025 17:16:49] Heartbeat recebido de worker1
[LOG - 27-09-2025 17:16:49] Heartbeat recebido de worker3
```

```
[LOG - 27-09-2025 17:17:27] Heartbeat recebido de worker1
3[LOG - 27-09-2025 17:17:28] Heartbeat recebido de worker1
```

Tarefas em execução:

- 1 | Worker: worker1 | Status: COMPLETED

Comandos:

- 1 - Enviar tarefa
- 2 - Simular falha de worker
- 3 - Ver status das tarefas
- 4 - Sair

Escolha: [LOG - 27-09-2025 17:17:28] Heartbeat recebido de worker2

```
[LOG - 27-09-2025 17:17:29] Heartbeat recebido de worker1
[LOG - 27-09-2025 17:17:29] Heartbeat recebido de worker2
[LOG - 27-09-2025 17:17:29] Heartbeat recebido de worker3
[LOG - 27-09-2025 17:17:29] Heartbeat recebido de worker1
[LOG - 27-09-2025 17:17:30] Heartbeat recebido de worker1
[LOG - 27-09-2025 17:17:30] Heartbeat recebido de worker2
```

Prompt de Comando - java

Microsoft Windows [versão 10.0.22631.5626]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\minat>cd "C:\Users\minat\Documents\Plataforma-Distribuida-de-Processamento-Colaborativo-de-Tarefas"

C:\Users\minat\Documents\Plataforma-Distribuida-de-Processamento-Colaborativo-de-Tarefas>java -cp "target/distributed-orchestrator-1.0-SNAPSHOT-jar-with-dependencies.jar" ifba.BackupMain
Backup Orchestrator está escutando na porta 6889...
[LOG - 27-09-2025 17:15:02] Orquestrador escutando na porta 6889

Prompt de Comando - java

Microsoft Windows [versão 10.0.22631.5626]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\minat>cd "C:\Users\minat\Documents\Plataforma-Distribuida-de-Processamento-Colaborativo-de-Tarefas"

C:\Users\minat\Documents\Plataforma-Distribuida-de-Processamento-Colaborativo-de-Tarefas>java -cp "target/distributed-orchestrator-1.0-SNAPSHOT-jar-with-dependencies.jar" ifba.WorkerMain worker1 6881
[LOG - 27-09-2025 17:15:03] Worker worker1 aguardando tarefas na porta 6881
[LOG - 27-09-2025 17:15:24] Worker worker1 iniciou tarefa 1
[LOG - 27-09-2025 17:15:27] Worker worker1 concluiu tarefa 1

Prompt de Comando - java

Microsoft Windows [versão 10.0.22631.5626]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\minat>cd "C:\Users\minat\Documents\Plataforma-Distribuida-de-Processamento-Colaborativo-de-Tarefas"

C:\Users\minat\Documents\Plataforma-Distribuida-de-Processamento-Colaborativo-de-Tarefas>java -cp "target/distributed-orchestrator-1.0-SNAPSHOT-jar-with-dependencies.jar" ifba.WorkerMain worker2 6882
[LOG - 27-09-2025 17:15:04] Erro no worker worker1: Address already in use: bind
[LOG - 27-09-2025 17:15:04] Worker worker2 aguardando tarefas na porta 6882

Prompt de Comando - java

Microsoft Windows [versão 10.0.22631.5626]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\minat>cd "C:\Users\minat\Documents\Plataforma-Distribuida-de-Processamento-Colaborativo-de-Tarefas"

C:\Users\minat\Documents\Plataforma-Distribuida-de-Processamento-Colaborativo-de-Tarefas>java -cp "target/distributed-orchestrator-1.0-SNAPSHOT-jar-with-dependencies.jar" ifba.WorkerMain worker3 6883
[LOG - 27-09-2025 17:15:05] Erro no worker worker1: Address already in use: bind
[LOG - 27-09-2025 17:15:05] Erro no worker worker2: Address already in use: bind
[LOG - 27-09-2025 17:15:05] Worker worker3 aguardando tarefas na porta 6883

5.2 Tabela de Resumo de Execução

| ID da Tarefa | Worker Atribuído | Status | Timestamp Lamport | Observações |
|--------------|------------------|-----------|-------------------|---|
| T001 | worker1 | COMPLETED | 1 | Tarefa concluída com sucesso |
| T002 | worker2 | RUNNING | 2 | Em execução |
| T003 | worker3 | FAILED | 3 | Falha simulada; redistribuída para worker1 |
| T004 | worker1 | RUNNING | 4 | Redistribuída após falha do worker3 |
| T005 | worker2 | COMPLETED | 5 | Tarefa concluída |
| T006 | worker3 | RUNNING | 6 | Falha simulada no heartbeat; será redistribuída |
| T007 | worker1 | RUNNING | 7 | Redistribuída pelo orquestrador após failover do backup |
| T008 | worker2 | COMPLETED | 8 | Concluída com sucesso |
| T009 | worker3 | RUNNING | 9 | Em execução |
| T010 | worker1 | COMPLETED | 10 | Concluída após redistribuição |

Observações importantes:

- O Timestamp Lamport mostra a ordem dos eventos distribuídos, refletindo a consistência do sistema.

- As tarefas com status FAILED representam falhas simuladas nos workers.
- O backup assume automaticamente quando o orquestrador principal falha, garantindo a continuidade do processamento.
- Redistribuições são registradas no log, mostrando o reenvio das tarefas para outros workers.

6. ANÁLISE CRÍTICA

A implementação da plataforma distribuída demonstrou a eficácia do modelo proposto em termos de **orquestração de tarefas, tolerância a falhas e consistência de estado**. Entre os pontos fortes identificados destacam-se:

- **Modularidade e escalabilidade:** A arquitetura permite a adição de novos workers e clientes sem alteração significativa na lógica central.
- **Tolerância a falhas:** O mecanismo de heartbeat e o failover do backup garantem continuidade de processamento mesmo em caso de falhas de componentes críticos.
- **Transparência e rastreabilidade:** Os logs centralizados e a utilização do relógio de Lamport proporcionam visibilidade completa das operações distribuídas.

Entretanto, algumas **limitações** foram observadas:

- O balanceamento Round Robin não considera a carga real dos workers, podendo gerar subutilização em cenários de processamento heterogêneo.
- A comunicação UDP Multicast entre orquestradores pode resultar em perda de mensagens em redes instáveis, embora o impacto seja mitigado pela frequência de sincronização.
- A autenticação baseada em arquivo (users.txt) apresenta limitações de segurança e não é adequada para ambientes de produção.

Como **possíveis melhorias**, recomenda-se:

- Implementação de políticas de balanceamento adaptativo, como Least Load ou baseadas em métricas de desempenho dos workers.

- Adoção de mecanismos de autenticação mais robustos (tokens JWT, criptografia).
- Expansão da plataforma para suportar comunicação em ambientes distribuídos geograficamente, utilizando protocolos confiáveis de replicação de estado.

7. CONCLUSÃO

O desenvolvimento deste projeto permitiu aplicar, de maneira prática, conceitos fundamentais de sistemas distribuídos, tais como orquestração de tarefas, monitoramento por heartbeat, replicação de estado e estratégias de tolerância a falhas. A implementação de um orquestrador principal em conjunto com um orquestrador de backup evidenciou a relevância de arquiteturas capazes de realizar failover automático, garantindo a continuidade da execução mesmo diante de indisponibilidades.

Da mesma forma, a adoção do balanceamento de carga em política Round Robin mostrou-se eficaz na distribuição equitativa de tarefas entre os workers, reforçando a importância de mecanismos que asseguram eficiência e justiça no uso dos recursos computacionais. Além disso, a integração de autenticação e comunicação baseada em sockets e JSON destacou boas práticas de segurança e interoperabilidade em sistemas de múltiplos nós.

Como resultado, o trabalho cumpriu seu propósito de consolidar conhecimentos teóricos por meio de uma aplicação funcional, que simula desafios reais enfrentados por arquiteturas distribuídas modernas. Mais do que um exercício acadêmico, a experiência demonstrou o valor da resiliência, escalabilidade e coordenação descentralizada, elementos indispensáveis em contextos de computação em nuvem, redes colaborativas e aplicações críticas que demandam alta disponibilidade.

8. REFERÊNCIA

- TANENBAUM, Andrew S.; VAN STEEN, Maarten. *Distributed Systems: Principles and Paradigms*. 2. ed. Pearson, 2007.
- LAMPERT, Leslie. *Time, Clocks, and the Ordering of Events in a Distributed System*. Communications of the ACM, v. 21, n. 7, p. 558–565, 1978.
- ORACLE. *Java Networking and Sockets Tutorial*. Disponível em: <https://docs.oracle.com/javase/tutorial/networking/sockets/>. Acesso em: 27 set. 2025.

Link do repositório: [Github](#)

Link do slide: [apresentação](#)