Plataforma Distribuída

Processamento Colaborativo de Tarefas

Projeto Final – Sistemas Distribuídos

IFBA - Campus Santo Antônio de Jesus

Apresentado por: Hildemar/Thiago/Kleberson

Professor: Felipe

Materia: Sistemas Distribuidos

Agenda da Apresentação

01

Introdução e Objetivos

Contextualização do projeto e metas estabelecidas

02

Fundamentação Teórica

Conceitos de sistemas distribuídos aplicados

03

Arquitetura do Sistema

Componentes e estrutura da plataforma

04

Implementação Técnica

Decisões de desenvolvimento e tecnologias

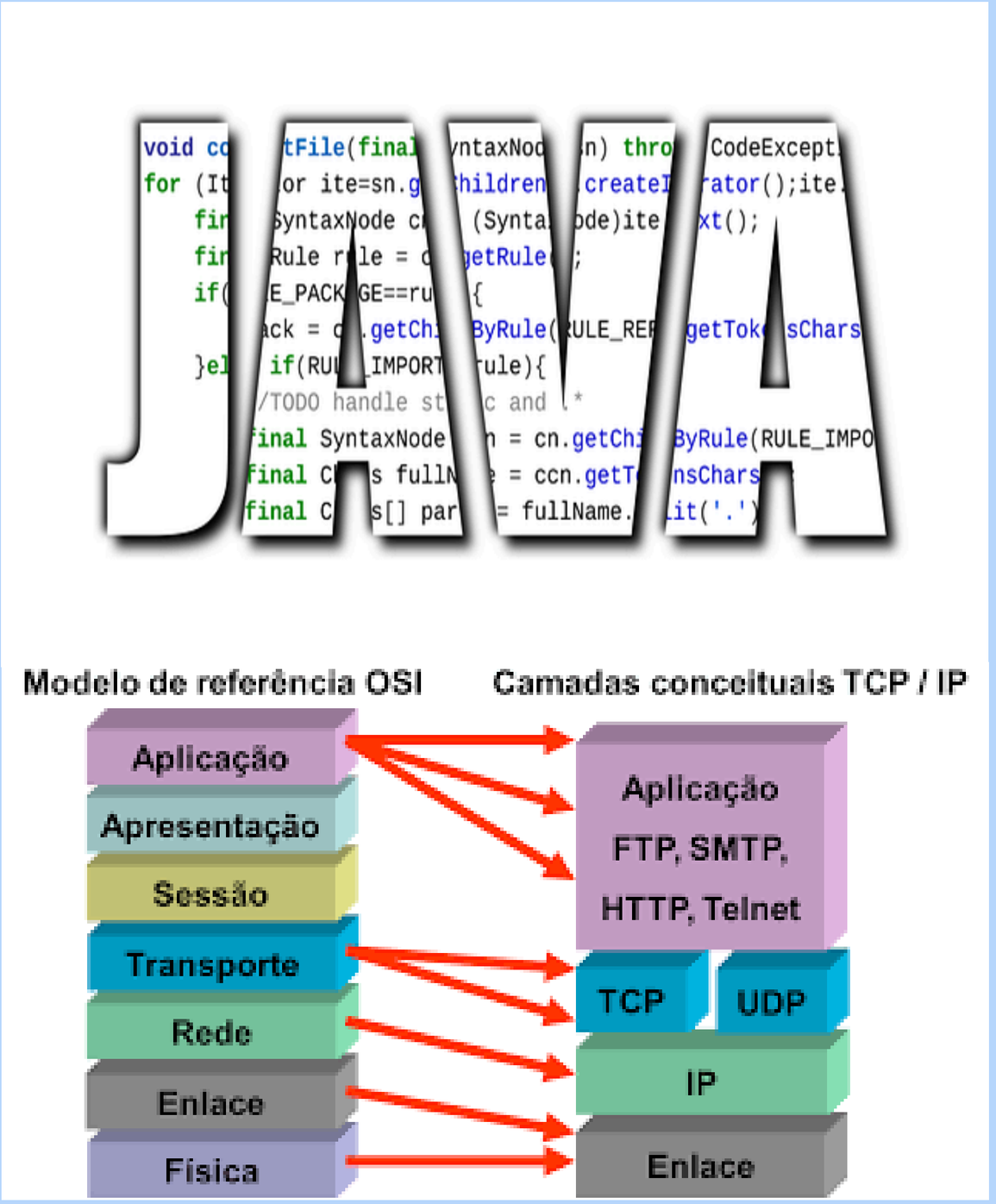
05

Resultados e Análise

Evidências de funcionamento e avaliação crítica

Introdução ao Projeto

Este trabalho apresenta o desenvolvimento de uma **Plataforma Distribuída De Processamento Colaborativo de Tarefas**, implementada como projeto final da disciplina Sistemas Distribuídos.



Objetivos Principais

- Receber e distribuir tarefas entre múltiplos nós Manter estado global consistente.
- Garantir recuperação automática de falhas.
- Implementar orquestração com backup.

Tecnologia: Java com sockets TCP/UDP puros, sem frameworks externos ou ferramentas REST.

Fundamentos de Sistemas Distribuídos

"Um sistema distribuído consiste em um conjunto de computadores independentes que se apresentam ao usuário como um sistema único e coerente" - Tanenbaum & Van Steen

Transparência

O sistema deve apresentar-se como uma unidade coesa, ocultando a complexidade da distribuição dos usuários

Compatilhamento

Recursos computacionais são compartilhados eficientemente entre múltiplos componentes

Tolerância a Falhas

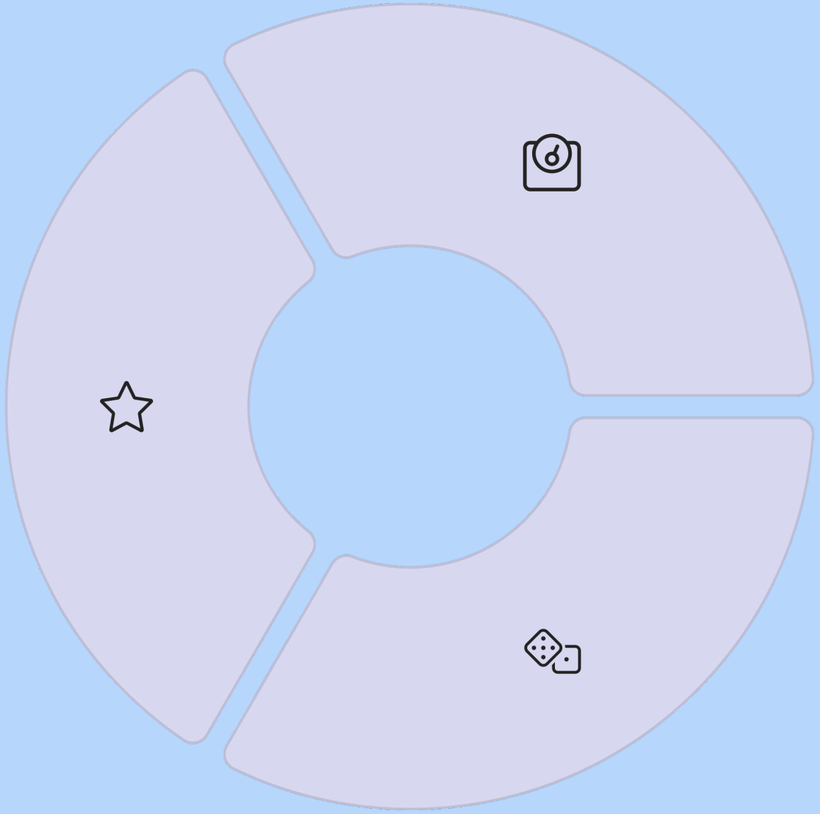
O sistema mantém funcionamento mesmo quando componentes individuais falham.

Estratégias de Balanceamento de Carga

Round Robin
Distribuição sequencial circular

-

(Escolha para o projeto)



Least Load
Atribuição ao nó menos sobrecarregado

Aleatória

Escolha sem critério determinístico

A política **Round Robin** foi adotada por sua simplicidade, previsibilidade e capacidade de garantir distribuição equitativa entre workers ativos.

Mecanismos de Tolerância a Falhas

Heart beat

Workersenviam sinais periódicos confirmando atividade ao orquestrador
principal

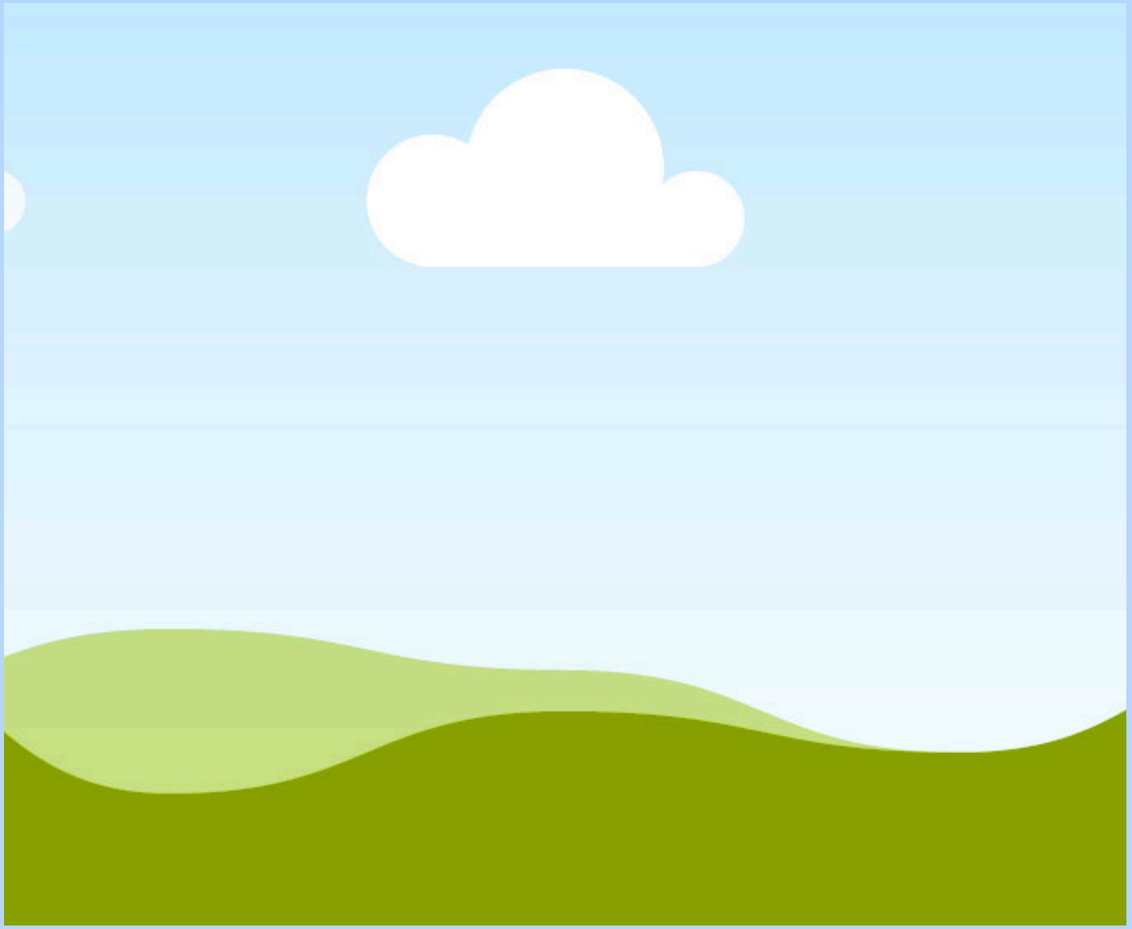
Failover Automático

Orquestrador backup assume controle automaticamente em caso de falha do principal

Redistribuição

Tarefas inacabadas são reatribuídas a outros nós ativos quando um worker falha

Relógios Lógicos de Lamport



Em sistemas distribuídos, a ausência de relógio global consistente exige mecanismos especiais de ordenação temporal.

Aplicação no Projeto

- Ordenação de submissões de tarefas
- Sincronização de atualizações de estado
- Garantia de consistência em operações de failover

Os timestamps lógicos asseguram relação causal entre eventos, prevenindo inconsistências durante redistribuição e failover.

Arquitetura do Sistema

Cliente

Submete tarefas e consulta status via TCP com o orquestrador principal

Orquestrador Principal

Coordena o sistema,balanceia carga e monitora workers

Orquestrador Backup

Mantém cópia do estadoviaUDP multicast, assume controle em falhas

Workers

Executam tarefas atribuídas e enviam heart beats periódicos

Arquitetura do Sistema

Estrutura do Projeto

Organizada em pacotes que separam responsabilidades:

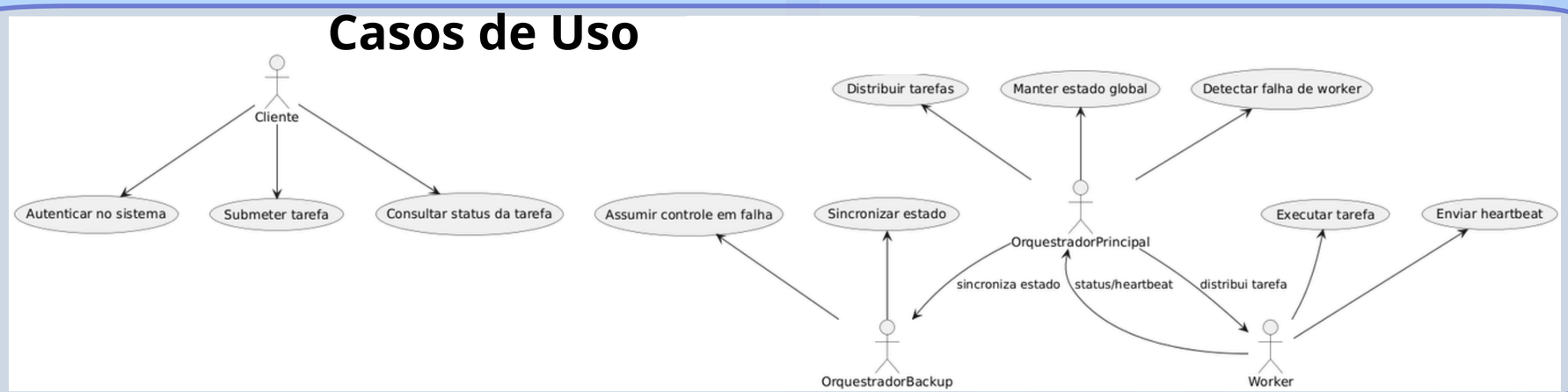
- **cliente –** interação com o usuário
- **modelo –** entidades e estados do sistema
- **orquestrador –** coordenação da execução distribuída
- **worker –** execução das tarefas
- **util –** funções auxiliares
- **scripts –** automação de compilação e execução

```
PLATAFORMADISTRIBUIDA
│
├── bin
│
├── src
│   ├── cliente
│   │   └── Cliente.java
│   │
│   ├── modelo
│   │   ├── EstadoGlobal.java
│   │   ├── Tarefa.java
│   │   └── Usuario.java
│   │
│   ├── orquestrador
│   │   ├── OrquestradorBackup.java
│   │   └── OrquestradorPrincipal.java
│   │
│   ├── util
│   │   └── RelogioLamport.java
│   │
│   └── worker
│       └── Worker.java
│
├── compile_and_run.sh
└── README.md
```

Tudo foi pensado para manter clareza, modularidade e facilitar a manutenção.

Diagramas UML

Casos de Uso

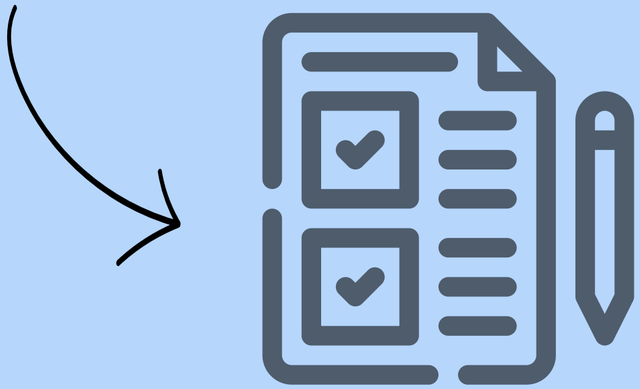


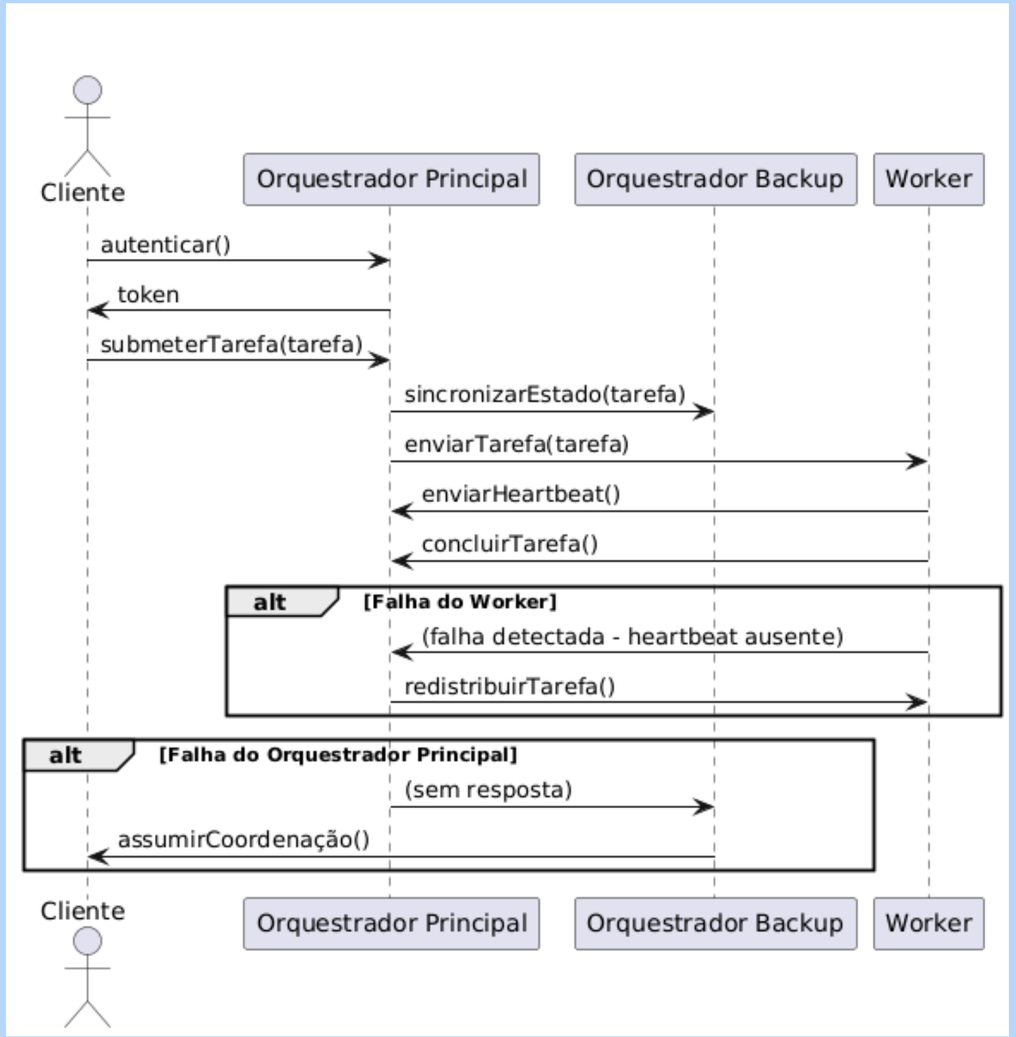
A ideia desse diagrama é mostrar de forma simples e clara quais são os papéis no sistema e o que cada um pode fazer. Ele ajuda a ter uma visão de alto nível, sem entrar nos detalhes técnicos, mas já deixando evidente que o sistema não depende apenas de um único ponto, já que existe a redundância com o orquestrador de backup e o mecanismo de detecção de falhas dos workers.

Diagramas UML

Sequencia

Nesse diagrama também aparecem os cenários de falha. Se um worker parar de responder, o orquestrador principal percebe pela ausência de heartbeat, marca esse worker como inativo e redistribui as tarefas dele para outro worker disponível. Já no caso do orquestrador principal falhar, o backup detecta a falta de comunicação, assume o papel de principal e continua o funcionamento sem que o cliente perca as informações das tarefas. Esse diagrama é importante porque mostra não só o funcionamento normal, mas também como o sistema reage em situações de erro.

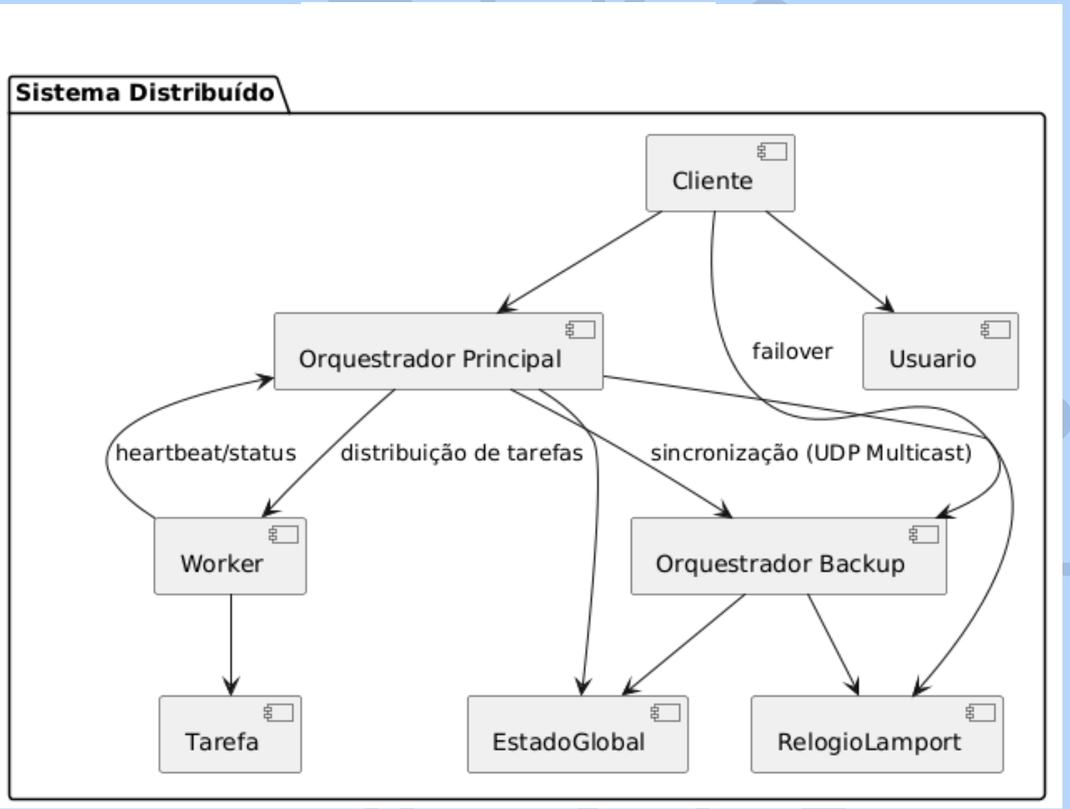




Diagramas UML

Componentes

Esse diagrama de componentes mostra a arquitetura do sistema, ou seja, quais são as partes que compõem ele e como essas partes se relacionam. O cliente se comunica com o orquestrador principal por meio de gRPC ou sockets TCP. O orquestrador principal e o backup se comunicam entre si usando UDP multicast para manter o estado sincronizado. O orquestrador principal também é quem envia as tarefas para os workers e recebe de volta os heartbeats e os resultados das execuções.



Padrões de Comunicação

Cliente Orquestrador Principal

TCP: Garante confiabilidade na submissão e consulta de tarefas

Orquestrador Principal Backup

UDPMulticast: Sincronização eficiente do estado global

Orquestrador Workers

Atribuição de tarefas e recebimento de heart beats

Padrões de Comunicação

Relatos de Execuções

1

```
[OrquestradorPrincipal] TCP server started on port 5000
[OrquestradorPrincipal] Worker registrado: worker1
[OrquestradorPrincipal] Worker registrado: worker2
[OrquestradorPrincipal] Worker registrado: worker3
[OrquestradorPrincipal] Cliente autenticado: junior
[OrquestradorPrincipal] Recebeu tarefa: t1,junior,10,PENDING
[OrquestradorPrincipal] Enviou tarefa t1 para worker worker1
[OrquestradorPrincipal] Recebeu tarefa: t2,junior,10,PENDING
[OrquestradorPrincipal] Enviou tarefa t2 para worker worker2
[OrquestradorPrincipal] Tarefa concluída: t1,junior,10,PENDING por worker worker1
[OrquestradorPrincipal] Recebeu tarefa: t3,junior,10,PENDING
[OrquestradorPrincipal] Enviou tarefa t3 para worker worker3
[OrquestradorPrincipal] Tarefa concluída: t2,junior,10,PENDING por worker worker2
[OrquestradorPrincipal] Tarefa concluída: t3,junior,10,PENDING por worker worker3
```

Cliente efetuando login para funções

Orquestrador principal dividindo tarefas entre Workers

2

```
Backup] Estado recebido via multicast: t1,junior,10,PENDING:worker1:RUNNING
Backup] Estado recebido via multicast: t1,junior,10,PENDING:worker1:RUNNING
Backup] Estado recebido via multicast: t1,junior,10,PENDING:worker1:RUNNING
Backup] Estado recebido via multicast: t1,junior,10,PENDING:worker1:RUNNING
Backup] Estado recebido via multicast: t2,junior,10,PENDING:worker2:RUNNING;t1,junior,10,PENDING:worker1:RUNNING
Backup] Estado recebido via multicast: t2,junior,10,PENDING:worker2:RUNNING
Backup] Estado recebido via multicast: t2,junior,10,PENDING:worker2:RUNNING
Backup] Estado recebido via multicast: t2,junior,10,PENDING:worker2:RUNNING
Backup] Estado recebido via multicast: t2,junior,10,PENDING:worker2:RUNNING;t3,junior,10,PENDING:worker3:RUNNING
Backup] Estado recebido via multicast: t3,junior,10,PENDING:worker3:RUNNING
Backup] Estado recebido via multicast: t3,junior,10,PENDING:worker3:RUNNING
Backup] Estado recebido via multicast: t3,junior,10,PENDING:worker3:RUNNING
Backup] Estado recebido via multicast: t3,junior,10,PENDING:worker3:RUNNING
```

```
[Backup] N?o recebeu estado do principal recentemente. Assumindo papel de Orquestrador Principal...
[Backup-as-Primary] TCP server started on port 5000
```

3

Backup de todo o processo

Caso o orquestrador principal cair, backup entra em ação

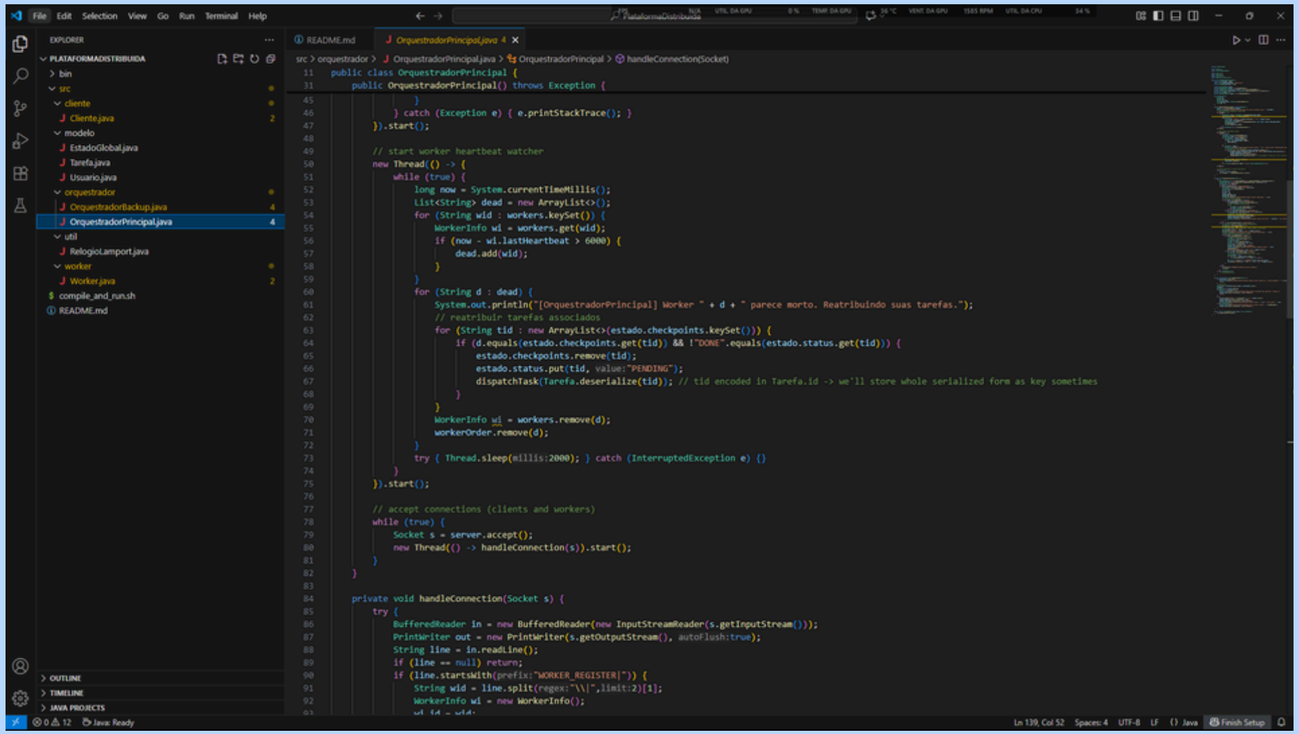
Conclusões e Perspectivas

Objetivos Alcançados

- Plataforma funcional com balanceamento Round Robin
- Tolerância a falhas com heartbeat e failover
- Autenticação básica implementada
- Uso efetivo de relógios lógicos de Lamport

Melhorias Futuras

- Balanceamento dinâmico baseado em métricas
- reais Criptografia TLS para comunicação segura
- Persistência em banco distribuído
- Containerização com Docker



> A plataforma demonstra com sucesso os conceitos fundamentais de sistemas distribuídos, servindo como base sólida para implementações mais complexas.