Politécnico de Viseu

Escola Superior de Tecnologia e Gestão de Viseu

Sistema Distribuído para Armazenamento de Documentos com FAISS e IPFS

Unidade Curricular de Sistemas Distribuídos

Licenciatura em Engenharia Informática

David Simão Borges 27431

José Maria Quental Arrais 23747

Patrícia Isabel Gaspar Oliveira 22525

Susana Daniela Tavares 27467

dezembro de 2025

Politécnico de Viseu
Escola Superior de Tecnologia e Gestão de Viseu

departamento de informática

Sistema Distribuído para Armazenamento de Documentos com FAISS e IPFS

Unidade Curricular de Sistemas Distribuídos

Licenciatura em Engenharia Informática

David Simão Borges 27431

José Maria Quental Arrais 23747

Patrícia Isabel Gaspar Oliveira 22525

Susana Daniela Tavares 27467

dezembro de 2025

Índice

1 Introdução

Este projeto implementa um sistema distribuído inovador que combina três tecnologias fundamentais para criar uma plataforma robusta de armazenamento e pesquisa descentralizada de documentos. O sistema integra IPFS (InterPlanetary File System) para armazenamento distribuído, RAFT para eleição de líder e coordenação entre nós, e FAISS para pesquisa semântica baseada em inteligência artificial.

O objetivo principal é permitir que múltiplos computadores (peers) trabalhem em conjunto formando uma rede descentralizada, onde documentos podem ser carregados, votados democraticamente pela comunidade, e pesquisados através de uma interface de pesquisa semântica avançada. O sistema garante tolerância a falhas através do consenso distribuído e eleição dinâmica de líder.

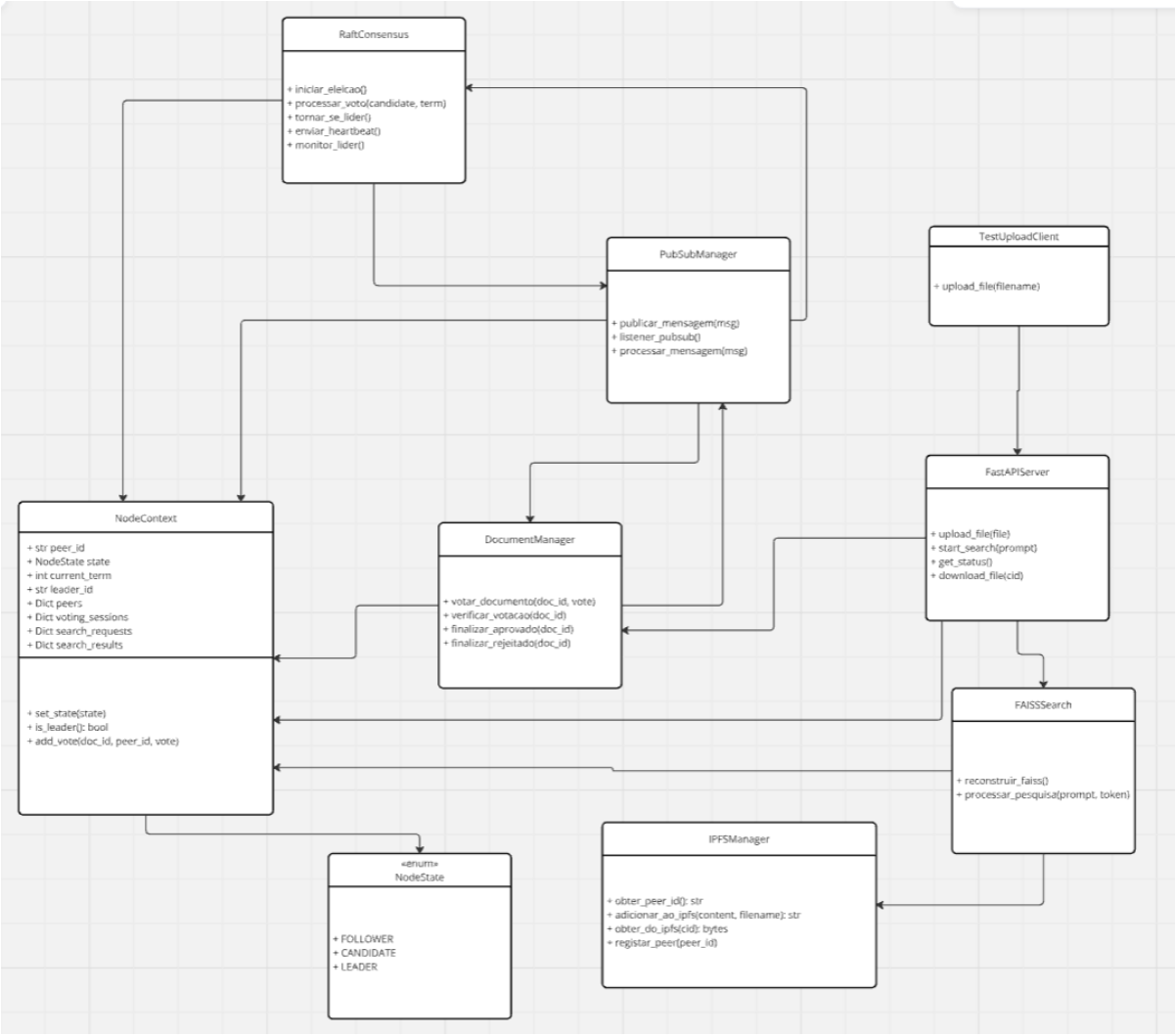
A principal inovação reside na combinação harmoniosa destas três tecnologias: enquanto o RAFT garante coordenação determinística e eleição de líder, o IPFS fornece armazenamento imutável e descentralizado, e o FAISS permite pesquisas semânticas eficientes em tempo real sobre documentos distribuídos.

2 Arquitetura da solução

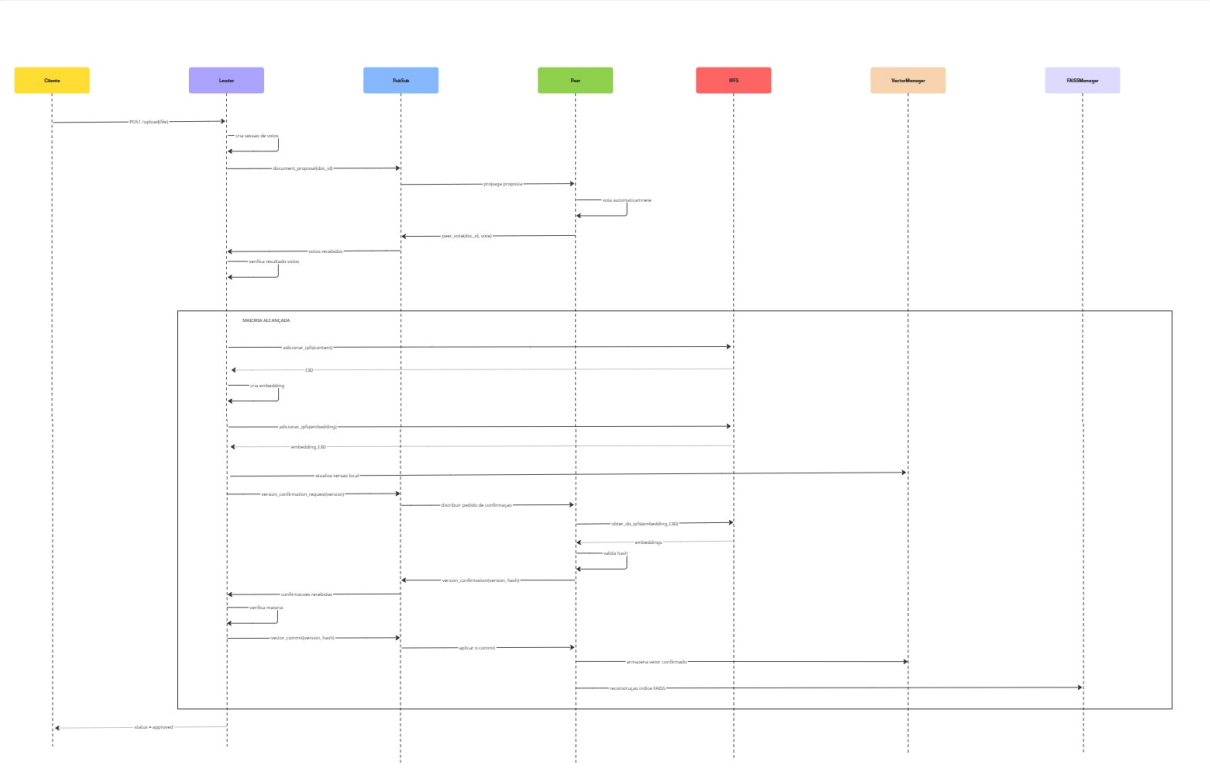
Para melhor vizualização, os diagramas podem ser acedidos no seguinte link:

https://miro.com/app/board/uXjVJixC3Lc=/?share_link_id=295364885643

2.1 Diagrama de Classes



2.2 Diagrama de sequência



3 Implementações

3.1 Gestão de estados RAFT

O ficheiro node.py define uma classe NodeContext que guarda todo o estado distribuído do nó: identidade (peer_id), estado RAFT (FOLLOWER, CANDIDATE, LEADER), current_term, voted_for, leader_id e timestamps do último heartbeat do líder. Esta classe também mantém estruturas partilhadas para sessões de votação de documentos, confirmações de versão, pesquisas em curso e resultados, protegidas por um RLock para garantir acesso thread-safe.

3.2 Loop de eleição e pedido de votos

A função iniciar_eleicao() coloca o nó no estado CANDIDATE, incrementa o current_term, regista o voto em si próprio e envia uma mensagem request_vote via PubSub. As funções processar_pedido_voto() e processar_resposta_voto() tratam os pedidos e respostas de voto, atualizando o term quando necessário e contando votos até atingir a maioria, momento em que tornar_se_lider() é chamada para assumir a liderança.

3.3 Heartbeats e deteção de falhas

A função enviar_heartbeat() envia periodicamente mensagens leader_heartbeat com o leader_id, term atual, número de documentos confirmados e propostas pendentes, ou peer_heartbeat quando o nó não é líder. Os heartbeats são usados em conjunto com timeouts (LEADER_TIMEOUT, PEER_TIMEOUT) para marcar peers inativos e disparar novas eleições quando o líder deixa de comunicar.

3.4 API HTTP do líder (FastAPI)

Quando um nó é eleito, tornar_se_lider() arranca o servidor FastAPI através de iniciar_servidor_http(), expondo endpoints HTTP em criar_aplicacao_fastapi(). Os principais endpoints são: POST /upload para criar propostas de novos documentos, POST /search e GET /search/{search_id} para gerir o ciclo completo de pesquisas semânticas, além de /status, /documents e /download/{cid} para monitorização e acesso a ficheiros.

3.5 Votação de documentos e Two-Phase Commit

No POST /upload, o líder guarda o ficheiro em pending_uploads, cria uma entrada em voting_sessions com os votos necessários e publica uma mensagem document_proposal. Depois de recolher uma maioria de votos de aprovação, o líder adiciona o ficheiro e os embeddings ao IPFS, calcula o hash da nova lista de documentos e usa as funções processar_pedido_confirmacao(), enviar_confirmacao_ao_lider() e enviar_commit() para implementar um protocolo em duas fases que distribui a nova versão pelos peers de forma consistente.

3.6 Pipeline de pesquisa semântica (peer executor)

O endpoint POST /search cria um search_id e um token, escolhe um peer em round-robin e regista a pesquisa em search_requests, podendo processar localmente em modo single-node. A função processar_pesquisa_faiss() (definida em node.py) carrega o índice FAISS, gera o embedding da prompt com SentenceTransformer, obtém os vizinhos mais próximos e guarda os resultados em search_results, que são depois expostos ao cliente via GET /search/{search_id}.

3.7 Garbage collector e tolerância a falhas

A função garbage_collector() corre em background e remove sessões de votação antigas, confirmações de versão expiradas e peers inativos, evitando acumulação de lixo e entradas desatualizadas. Em paralelo, o script check_setup.py fornece uma ferramenta de diagnóstico que verifica versões de Python, módulos instalados, estado do IPFS, PubSub, mDNS, portas, diretórios e conectividade de rede, permitindo garantir que o ambiente está pronto antes de arrancar o node.py.

4 Conclusão

A solução cumpre o objetivo de criar um sistema distribuído tolerante a falhas, capaz de armazenar documentos no IPFS, fazer pesquisa semântica com FAISS e coordenar tudo via RAFT, garantindo consistência e liderança única na rede.

As principais limitações identificadas são dependência de uma rede estável, ausência de autenticação forte de utilizadores, alguma complexidade na gestão de estados distribuídos e falta de testes exaustivos em cenários reais de grande escala.

Como melhorias futuras, poderia se apostar em reforço de segurança (tokens mais robustos e autenticação), otimização de desempenho (caching, balanceamento mais avançado), monitorização e logging mais completos, e expansão de funcionalidades como filtros avançados de pesquisa e regras mais inteligentes para votação de documentos.