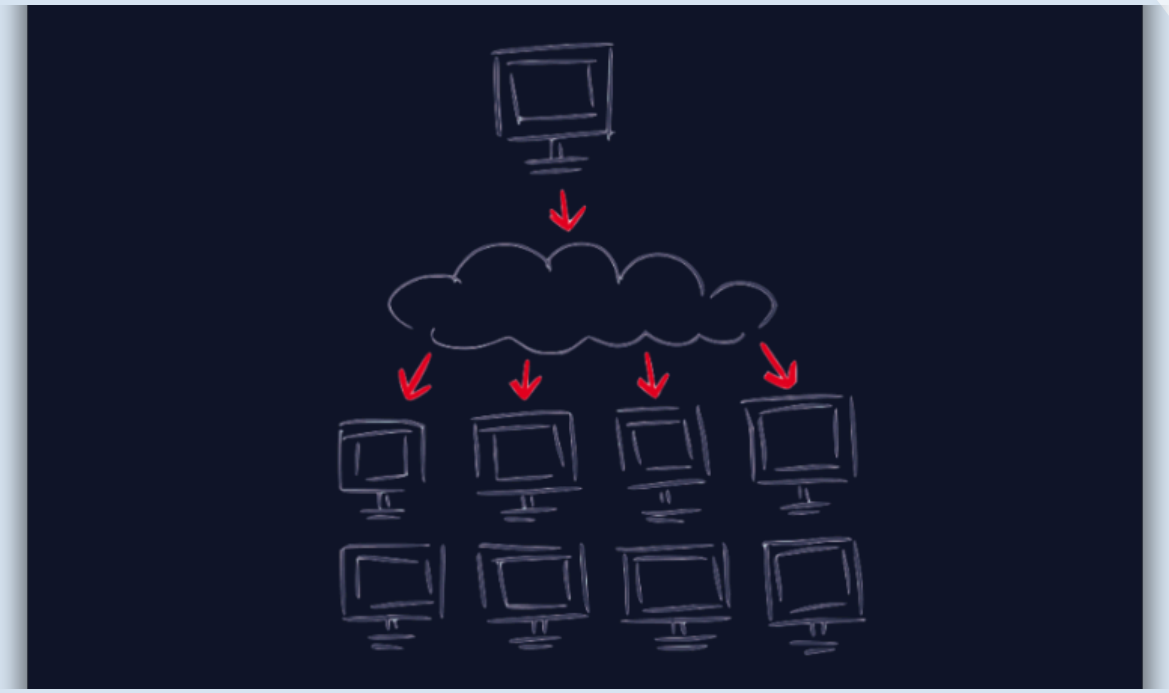
1 2 9 0 FACULDADE DE CIÊNCIAS E TECNOLOGIA UNIVERSIDADE D COIMBRA

dei departamento de engenharia informática

GOOGOL

Motor de pesquisa distribuído



25 de março de 2025

Diogo Costa, nº 2022213506
José Frota, nº 2022214661
Luana Carolina Reis, nº 2022220606

ÍNDICE

01. INTRODUÇÃO

O projeto *Googol* consiste num motor de pesquisa desenvolvido seguindo os princípios fundamentais dos sistemas distribuídos modernos, com uma arquitetura baseada em RMI/RPC. Foi concebido para oferecer funcionalidades completas de indexação e pesquisa *web*, com especial ênfase na tolerância a falhas e processamento paralelo. A arquitetura adotada reflete uma abordagem prática aos desafios de distribuição de dados, balanceamento de carga e recuperação após falhas, mantendo ao mesmo tempo uma *interface* simples para o utilizador final.
O sistema é composto por quatro componentes principais:

Componente	Descrição
Gateway	Servidor intermédio
Barrels	Armazenamento do índice invertido, replicado
Downloaders	*Crawlers* que processam páginas *web*
Client	Interface do utilizador

02. ARQUITETURA DO SISTEMA

Funcionamento

O sistema *Googol* organiza-se nestes quatro componentes principais, que comunicam entre si através de Java RMI, formando uma arquitetura do tipo cliente-servidor multicamada.

A *Gateway* serve como ponto central de coordenação, recebendo todos os pedidos dos clientes e distribuindo-os pelos diversos *Barrels*.

Estes últimos armazenam o índice invertido, particionado alfabeticamente em duas metades: A-M e N-Z, com cada partição tendo réplicas idênticas, de forma a garantir redundância.

Os Downloaders operam como workers paralelos, responsáveis pelo crawling efetivo das páginas web. Utilizam a biblioteca “Jsoup” para extrair não apenas o conteúdo textual, mas também os metadados mais relevantes, como títulos e descrições, bem como todos os hyperlinks presentes em cada página.

O Client oferece uma interface de linha de comandos simples mas funcional, permitindo tanto a pesquisa por termos como a administração básica do sistema.

A comunicação entre estes componentes segue um padrão bem definido. Quando um utilizador adiciona uma nova URL através do *Client*, este pedido é encaminhado para a *Gateway*, que por sua vez coloca o URL numa *queue* central. Os *Downloaders*, a trabalhar em paralelo, consomem desta fila, processam as páginas correspondentes e atualizam os *Barrels* através de chamadas RMI.

Um mecanismo de *reliable multicast* garante que todas as réplicas de cada partição recebam as mesmas atualizações.

1. Gateway

- Recebe pedidos de *Client* (pesquisas, adição de URLs).
- Distribui pesquisas pelos *Barrels* (A-M ou N-Z), por *Round-Robin*.
- Implementa *failover*: se um *Barrel* falhar, usa o redundante.

2. Barrels (A-M e N-Z)

- Armazenam o índice invertido (HashMap<String, Set<Page>>).
- Replicação via *reliable multicast* (atualizações são propagadas entre *peers*).
- Partição do índice, onde cada *Barrel* cobre metade do alfabeto, A-M ou N-Z.

3. *Downloaders*

- Processam URLs em paralelo, usando *jsoup*.
- Extraem texto, links e metadados (título, descrição).
- Atualizam os B*arrels* via RMI.

4. *Client*

- *Interface* simples para pesquisas e administração.

Diagrama de Componentes

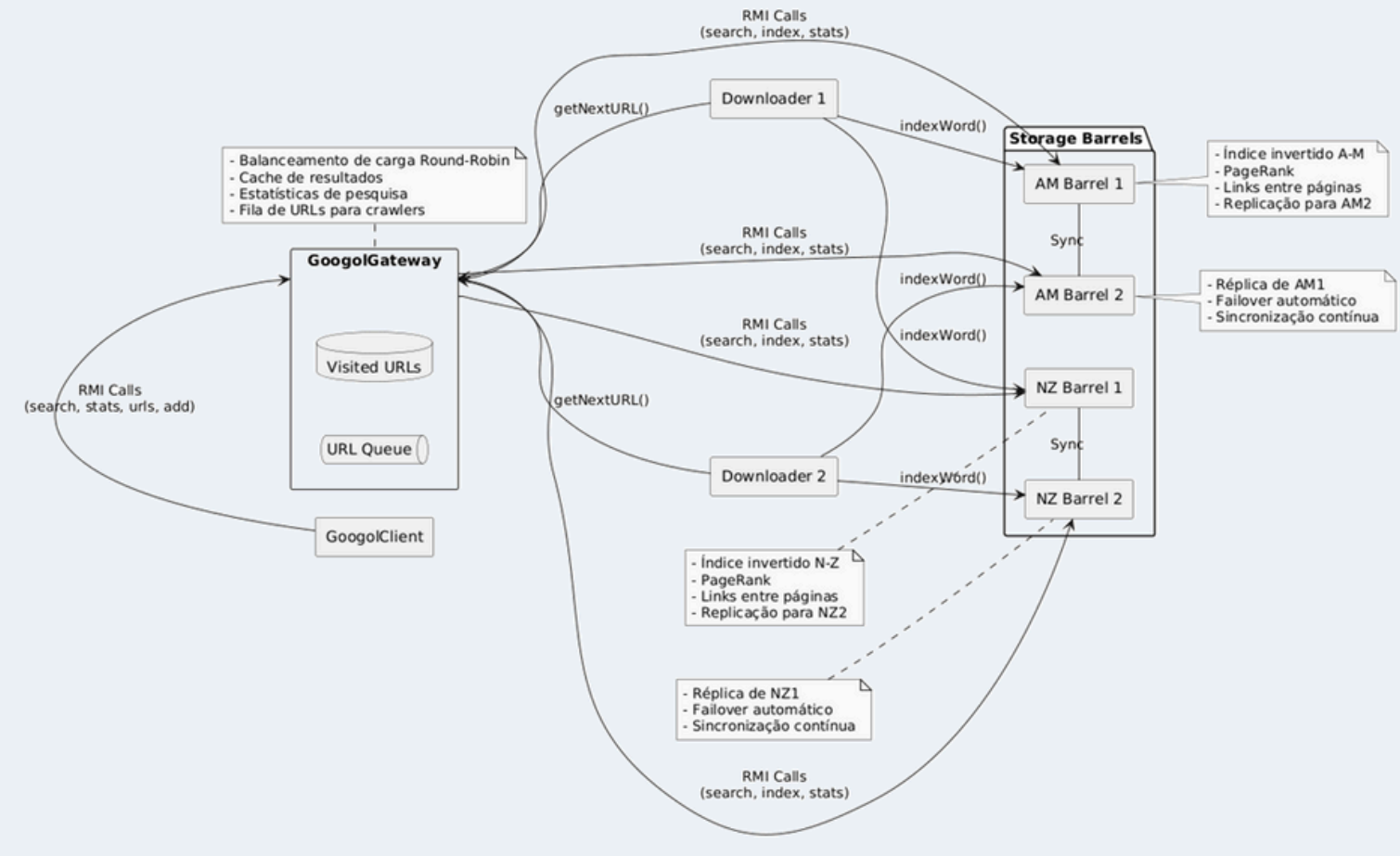


Fig. 1 - Diagrama UML do sistema implementado

03. TOLERÂNCIA A FALHAS E REPLICAÇÃO

A robustez do sistema *Googol* manifesta-se principalmente na sua abordagem à tolerância a falhas. Cada *Barrel* possui uma réplica idêntica que mantém exatamente os mesmos dados. Esta duplicação permite que o sistema continue operacional mesmo quando um dos servidores de armazenamento falha. A *Gateway* implementa um algoritmo inteligente de seleção de *Barrels* que deteta automaticamente falhas e redireciona os pedidos para uma réplica disponível.

O processo de replicação entre *Barrels* foi cuidadosamente desenhado para garantir consistência eventual, mesmo em cenários de falha. Quando um *Downloader* indexa nova informação, esta é primeiro enviada para o *Barrel* primário correspondente ao intervalo alfabético da palavra.

O *Barrel* primário assume então a responsabilidade de propagar esta atualização para o seu par, através de um protocolo de *reliable multicast* implementado sobre RMI. Caso o *Barrel* secundário esteja temporariamente indisponível, as atualizações são armazenadas numa fila persistente e sincronizadas assim que a conexão for restabelecida.

Para além da replicação em memória, os *Barrels* implementam um mecanismo de persistência em disco, que grava periodicamente o estado completo do índice invertido. Este ficheiro permite que um *Barrel* recupere rapidamente o seu estado após uma reinicialização, sincronizando-se depois com o seu par para garantir que não perdeu nenhuma atualização durante o período de inatividade. A combinação destas técnicas proporciona uma solução completa e que mantém a disponibilidade do sistema, mesmo perante falhas imprevistas.

Replicação em *Barrels*

- Estratégia: *Primary-Backup* (cada *Barrel* tem um par redundante).

- *Reliable Multicast*:
  - Quando um *Downloader* indexa uma palavra/URL, notifica todos os *Barrels* do mesmo tipo (A-M ou N-Z).
  - Se um *Barrel* falhar, as atualizações são armazenadas numa fila e sincronizadas quando recupera.

Failover na *Gateway*

Se um *Barrel* (exemplo: A-M1) não responder, a *Gateway* redireciona o pedido para o seu par (A-M2).

Recuperação de Estado

- Os *Barrels* gravam o índice em ficheiros (“index_storage_barrel.dat”).
- Ao reiniciar, carregam o último estado válido.

04. FUNCIONALIDADES IMPLEMENTADAS

O motor de pesquisa Googol implementa um conjunto abrangente de funcionalidades que cobrem todos os requisitos essenciais de um sistema de pesquisa moderno. O processo de indexação começa quando um utilizador introduz manualmente um URL, através da *interface* do *Client*. Este URL é adicionado a uma fila central e subsequentemente processado por um dos *Downloaders* disponíveis, que extrai não apenas o conteúdo textual da página mas também todos os links nela contidos, os quais são por sua vez adicionados à fila, para processamento futuro.

As pesquisas por termos suportam múltiplas palavras chave e devolvem apenas as páginas que contêm todas as palavras pesquisadas. O sistema implementa ainda uma função de paginação, que apresenta os resultados agrupados de dez em dez, tal como é comum nos motores de pesquisa comerciais (Google, Firefox, etc). A ordenação por relevância baseia-se num algoritmo simples de *PageRank* que considera o número de links que apontam para cada página: quanto mais *links* recebe uma página, mais alta irá aparecer nos resultados. Outra funcionalidade particularmente útil é a capacidade de consultar todas as páginas que contêm *links* para um determinado URL. Isto não só suporta o cálculo do *PageRank* como também fornece aos utilizadores informação valiosa sobre a rede de ligações entre páginas. O sistema inclui ainda uma consola de administração que apresenta estatísticas em tempo real sobre o estado do sistema, incluindo o tamanho do índice, o número de URLs pendentes, o tempo médio de resposta e o "top 10" de pesquisas mais frequentes.

Funcionalidade	Descrição
Indexação de URLs	*Clients* adicionam URLs manualmente; *Downloaders* processam-nos.
Pesquisa por termos	Devolve páginas que contêm todas as palavras pesquisadas.
Ordenação por *PageRank*	Resultados ordenados pelo número de links recebidos.

Funcionalidade	Descrição
Consulta de *links* para uma página	Lista todas as páginas que apontam para um URL específico.
Estatísticas em tempo real	Mostra tamanho do índice, *Barrels* ativos e fila de URLs pendentes.

05. DISTRIBUIÇÃO DE TAREFAS

Diogo Costa	Exercício 1, 2, 7 e Redundância
José Frota	Exercício 1, 2, 5 e 6
Luana	Exercício 1, 2, 3 e 4

Nota: Todos os elementos do grupo testaram e foram ajustando a solução.

06. TESTES REALIZADOS

A validação do sistema *Googol* envolveu uma quantidade bastante abrangente de testes, concebidos para verificar não apenas a correção funcional mas também a resiliência perante falhas. Os testes funcionais confirmaram que todas as operações básicas (adição de URLs, indexação automática, pesquisa por termos e consulta de links) funcionam conforme o esperado, em condições normais de operação.

Os testes de falha demonstraram a robustez do sistema quando confrontado com condições adversas. Simulações de falha de *Barrels* individuais confirmaram que a *Gateway* deteta automaticamente essa falha e redireciona os pedidos para a réplica correspondente, sem interrupção perceptível do serviço pela parte do utilizador. Testes mais elaborados, onde múltiplos componentes eram desligados simultaneamente, validaram os mecanismos de recuperação e persistência em disco.

A avaliação de desempenho focou-se no processamento paralelo de URLs pelos múltiplos *Downloaders* e na capacidade do sistema de lidar com carga crescente. Mesmo com centenas de URLs na fila de processamento, o sistema manteve um *throughput* constante, demonstrando a eficácia da abordagem paralela. Testes em ambiente distribuído, com componentes a executar em máquinas diferentes, confirmaram ainda a correta configuração da comunicação RMI através da rede.

Cenário	Resultado
• Adicionar URL manualmente	• URL é indexado e aparece em pesquisas ✅
• Pesquisa com múltiplos termos	• Devolve interseção de páginas relevantes ✅
• Falha de um *Barre*l AM	• *Gateway* usa *Barrel* AM2 sem interrupção ✅
• *Downloader* processa *links*	• *Links* são extraídos e adicionados à fila ✅
• Recuperação após *crash*	• *Barrel* recarrega índice do ficheiro ✅

07. CONCLUSÕES

A conclusão da primeira meta do projeto demonstra uma abordagem robusta e escalável, preparada para futuras expansões de componentes e funcionalidades. A arquitetura do sistema foi projetada com ênfase em soluções que garantem a operação ininterrupta do motor de pesquisa, especialmente no que diz respeito ao mecanismo de *failover* dos componentes críticos. Além disso, a sua clara separação de responsabilidades entre Gateway, Barrels e Downloaders, mostrou-se flexível e fácil de manter, ao mesmo tempo que permitiu manter um bom desempenho.

A solução final demonstra como conceitos teóricos como tolerância a falhas, consistência de dados e processamento paralelo podem ser efetivamente aplicados num sistema real. Todos os requisitos funcionais da *checklist* foram implementados com sucesso, incluindo a indexação recursiva de URLs, a pesquisa por múltiplos termos e a ordenação por relevância.

O *Googol* passou em todos os testes realizados, estando apto a uso. Para além disso, o programa foi testado com sucesso entre duas máquinas diferentes, mostrando assim que a implementação do RMI foi bem aplicada e configurada no projeto.

Esta meta inicial estabelece uma base sólida para o desenvolvimento futuro do sistema. A próxima fase prevê a implementação de um servidor *web* que, por meio de uma API, facilitará a integração entre o servidor RMI e a interface *web*, expandindo assim as capacidades do sistema distribuído.