Relatório Meta Orientadora Sistemas Distribuídos - 2024/2025

Bruna Dewes - uc2024243221@student.uc.pt
Cecília Quaresma - uc2024245307@student.uc.pt
Heloísa Centenaro - uc2024246775@student.uc.pt

1. Introdução

Este relatório apresenta o desenvolvimento do projeto sobre um mini motor de busca, semelhante ao Google, Frog e demais similares, que visa compreender e implementar os princípios fundamentais por trás de um motor de pesquisa.

Tal projeto envolve a criação de um sistema capaz de indexar e armazenar informações de páginas da *web* e suas respectivas *URLs* e, para isso, foram criados programas como *Downloaders*, *Web Crawlers*, *Barrels*, uma *Gateway* e um Cliente, nos quais foram aplicados conceitos de indexação, algoritmos de busca, servidores e estruturação de dados.

Além da implementação técnica, o relatório discute os desafios encontrados, as decisões tomadas no decorrer do projeto e alguns dos resultados obtidos. Com isso, busca-se demonstrar a importância dos motores de busca e explorar suas aplicações práticas.

Link do repositório: https://github.com/cqcoding/SistemasDistr/tree/main.

2. Arquitetura de Software

O diagrama abaixo demonstra de forma visual a organização do projeto e suas conexões.

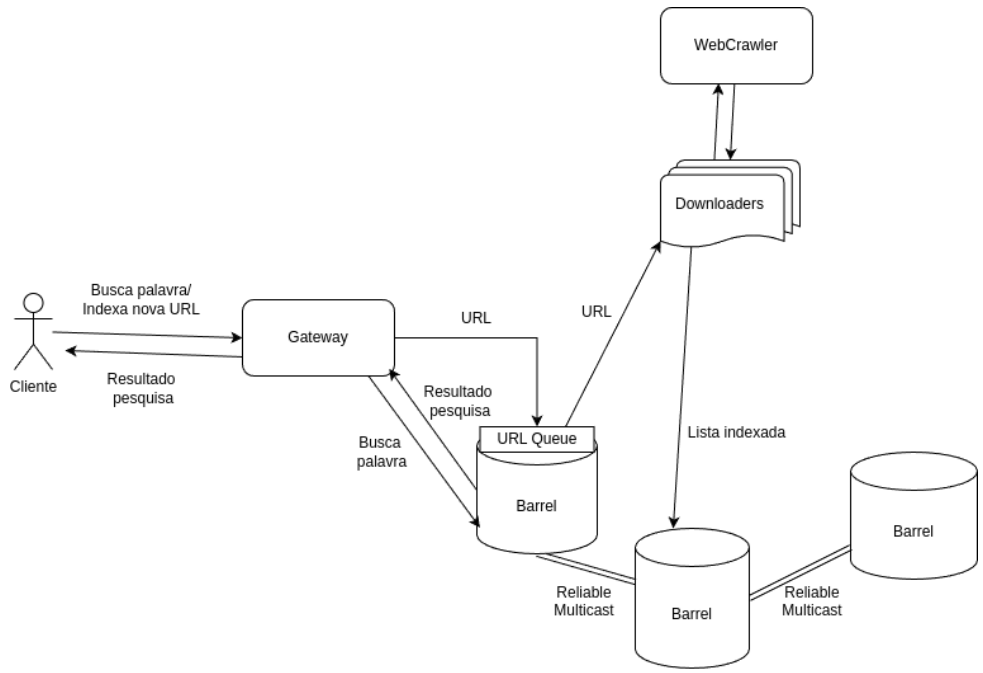

Fig 1: Arquitetura do Projeto

3. Decisões de Projeto

Na linguagem de programação optou-se por utilizar Java no ambiente Visual Studio Code, juntamente com uma arquitetura cliente-servidor, onde os principais módulos foram: Downloaders (responsáveis por coletar páginas da web e extrair as informações), Barrels (com servidores de indexação que armazenam e organizam os dados coletados), Gateway (com uma interface que recebe consultas da classe Cliente e retorna os resultados), Web Crawler (percorre a web e coleta as páginas).

Para a comunicação entre os módulos citados foi usado RMI (Remote Method Invocation), multicast fiável para garantir a replicação de informação entre os Barrels e balanceamento de carga a fim de distribuir os pedidos de indexação entre os Barrels e evitar sobrecarga.

Em questão de processamento e indexação foi decidido usar Jsoup para processar o HTML e coletar o conteúdo vindo das páginas web e método para remoção de *stop words*.

Além disso, como estratégia de busca utilizamos índices invertidos, onde as palavras são indexadas junto às URLs.

Optamos pelos testes de integração para validarmos as conexões remotas, os métodos RMI, a indexação e a sincronização dos dados. Ainda, realizamos diversos testes manuais para verificarmos as funcionalidades como a coleta de URLs, a indexação, as buscas, o retorno das pesquisas e das estatísticas.

De modo geral, houve dificuldades na gestão dos códigos, na implementação de algumas funcionalidades e na conexão cliente-servidor em diferentes máquinas, haja vista que foi o primeiro contato com este tipo de sistema e de projeto.

4. Downloaders e Barrels

A classe Downloader é responsável por baixar e processar URLs para indexação e se conecta a um dos servidores Barrel disponíveis, escolhendo aleatoriamente entre uma lista de URLs de servidores Barrel. A classe inicializa conjuntos para armazenar URLs processadas, palavras processadas, contagem de palavras e *stop words*, as quais são carregadas de um arquivo chamado **stopwords.txt**, onde contém palavras que são ignoradas durante o processamento de texto.

O método **executar** é um loop contínuo que busca URLs do servidor Barrel, processa o conteúdo da página e indexa palavras. URLs já processadas também são ignoradas, enquanto isso, as novas URLs encontradas na página são enviadas de volta ao servidor Barrel para futura indexação.

Palavras são extraídas do texto da página, convertidas para minúsculas e verificadas contra a lista de *stop words*. Palavras que não sejam *stop words* são contadas e, se não foram processadas anteriormente, são indexadas junto com a URL

correspondente. O método **atualizarStopWords** atualiza o arquivo de *stop words* com palavras que aparecem frequentemente. URLs e palavras processadas são salvas em arquivos para garantir que o estado seja mantido entre execuções.

A classe BarrelServer implementa a interface **InterfaceBarrel** e atua como um servidor que gerencia a indexação de URLs. Cada instância de BarrelServer possui um nome único e inicializa estruturas de dados para armazenar URLs indexadas e uma fila de URLs a serem processadas. As URLs previamente indexadas são carregadas de um arquivo chamado **urlsIndexados.txt**.

URLs são indexadas associando palavras a URLs em um mapa. O método **indexar_URL** adiciona uma URL ao índice e salva o estado no arquivo. O método pesquisar permite buscar URLs associadas a uma palavra específica. URLs são adicionadas a uma fila para processamento futuro. Métodos como **get_url** e **put_url** gerenciam a adição e remoção de URLs da fila.

O servidor pode sincronizar dados com outros servidores Barrel, garantindo que todos tenham o mesmo conjunto de dados indexados.

Métodos como **getNomeBarrel** e **getTamanhoIndice** fornecem informações sobre o estado do servidor, como o nome e o tamanho do índice.

Essas classes trabalham juntas para criar um sistema distribuído de indexação de URLs, onde os Downloaders coletam e processam dados, enquanto os Barrel Servers armazenam e gerenciam o índice de URLs.

5. Funcionamento RMI

O Java RMI permite que cada componente registre e exponha seus métodos para serem acessados remotamente. O funcionamento ocorre da seguinte forma:

- As interfaces remotas, como **InterfaceBarrel** e **InterfaceGatewayServer**, definem os métodos que podem ser chamados remotamente. Essas interfaces estendem **java.rmi.Remote** e os métodos lançam **RemoteException.**
- **GatewayServer:** Implementa a interface **InterfaceGatewayServer** e estende **UnicastRemoteObject**, o que permite que seus objetos sejam exportados automaticamente para chamadas remotas. Ele gerencia a conexão com os Barrels e realiza operações como indexação de URLs e pesquisa.
- **Downloader:** Conecta-se a um Barrel aleatório para baixar e processar URLs, enviando novas URLs para indexação.
- Cada Barrel é registrado no **RMI Registry**, permitindo que outros componentes possam localizá-lo;
- O GatewayServer busca dinamicamente os Barrels registrados para estabelecer comunicação.

No sistema, o Java RMI permite a comunicação entre os diferentes componentes distribuídos, executando operações remotas de maneira transparente. A implementação do Reliable Multicast e do monitoramento dos Barrels fortalece a confiabilidade e consistência do sistema, evitando falhas na indexação e recuperação de dados.

6. Distribuição de Tarefas

Inicialmente, a distribuição das tarefas foi feita da forma descrita abaixo, entretanto, no decorrer do projeto, todas as integrantes ficaram envolvidas com um pouco de cada tópico e contribuindo nos códigos.

Bruna Dewes: Gateway + Estatísticas em tempo real + Cliente

Implementa a Gateway, que recebe requisições e distribui para os Barrels.
Cuida da comunicação RPC/RMI entre Gateway e Barrels.
Desenvolve o cliente RPC/RMI usado pelos utilizadores para aceder às funcionalidades do Googol.
Desenvolve as estatísticas em tempo real, garantindo atualização e exibição de métricas.

Cecília Quaresma: Barrels + Storage Barrels + Multicast Fiável

Implementa os Storage Barrels, que armazenam os dados indexados.
Desenvolve o Reliable Multicast, garantindo replicação e consistência entre os Barrels.
Gerencia a fila de URLs, garantindo que novos links sejam explorados de forma eficiente.

Heloísa Centenaro: Web Crawler + Downloaders + Fila de URLs

Implementa os Downloaders (Web Crawlers) para baixar e processar páginas.
Gerencia a indexação invertida e mantém a estrutura de dados para buscas eficientes.
Cuida da extração de texto e links usando o Jsoup.

7. Testes

Id	Teste	Descrição	Critério de Aprovação	Status
T01	Conexão com o BarrelServer	Verificar se a conexão com o servidor RMI é estabelecida corretamente.	Conexão estabelecida sem exceções.	PASS
T02	indexar_URL	Testar se a URL é indexada corretamente no servidor.	A URL "https://example.com" deve ser indexada sem erros e uma mensagem de confirmação deve ser exibida.	PASS
T03	pesquisar	Verificar se a pesquisa	A pesquisa pela	PASS

		retorna resultados esperados	palavra "Anistia" deve retornar uma lista de resultados sem erros.	
T04	estaAtivo	Verificar se o servidor está ativo.	Retorna true se o servidor estiver ativo.	PASS
T05	getIndexados	Testar se os dados indexados são retornados corretamente.	Dados retornados correspondem aos dados indexados.	PASS
T06	sincronizarDados	Verificar se a sincronização de dados entre barrels funciona corretamente.	Dados sincronizados corretamente entre servidores.	PASS
T07	Conexão com GatewayServer	Testa se a aplicação consegue se conectar ao GatewayServer via RMI.	A conexão deve ser estabelecida sem lançar exceções de RemoteException ou NotBoundException.	PASS
T08	Downloader	Testa se o Downloader faz a indexação sem URLs duplicadas	O cliente deve receber uma lista com as palavras e suas respectivas URLs sem duplicações	FAIL

8. Bibliografia

ARTIGO ONLINE:

PATEL, N. (2025). *Web Crawler: O que é e como funciona?*. Disponível em https://neilpatel.com/br/blog/web-crawler/.

WHITEPRESS. (2025). *Motores de Busca: Como funcionam?*. Disponível em https://www.whitepress.com/pt/blog/6012/motores-de-busca.

DIGITALOCEAN. (2025). *Como fazer crawling em uma página web com Scrapy e Python 3*. Disponível em https://www.digitalocean.com/community/tutorials/como-fazer-crawling-em-uma-pagina-web-com-scrapy-e-python-3-pt.

DOCUMENTO ACADÊMICO:

RODRIGUES, M. A. dos S. (2016). *Sistema de recomendação baseado em web scraping para recuperação de informação personalizada*. Dissertação de Mestrado. Universidade Nova de Lisboa. Disponível em https://run.unl.pt/bitstream/10362/27885/1/Rodrigues_2016.pdf.

LIVRO DIDÁTICO:

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T.; BLAIR, G. (2012). *Distributed Systems: Concepts and Design*. 5.ª ed. Boston: Addison-Wesley.

SITE DE SOFTWARE:

JSOUP. (2025). *Download da biblioteca Jsoup*. Disponível em https://jsoup.org/download.

VÍDEO DO YOUTUBE:

CANAL: Cláudio Rodolfo Sousa de Oliveira. (2025). *Programação Distribuída - RMI com Java*. YouTube. Disponível em https://www.youtube.com/watch?v=zqAyjWgJ-lw.

LINK DO REPOSITÓRIO DO PROJETO NO *GITHUB*:

https://github.com/cqcoding/SistemasDistr/tree/main