UNIVERSIDADE D
COIMBRA

Googol Search Engine - Relatório

Sistemas Distribuídos - Meta 2

Marcelo Gomes Nº2021222994
Pedro Brites Nº2021226319

Índice

1. Arquitetura

A página inicial do nosso motor de busca (**Figura 1**) é bastante simples.

Figura 1: Página inicial

Temos uma caixa de texto com dois botões. O mais à esquerda serve para mudar o tipo de pesquisa. A pesquisa pode ser de um conjunto de palavras ou então de um url para obter os urls que apontam para ele. Após efetuarem a pesquisa os utilizadores são redirecionados para a página de resultados. O botão à direita direciona para a página de administração. Já a caixa de texto mais abaixo é utilizada para enviar urls para indexar.

Na página dos resultados de pesquisa são apresentados os resultados paginados de 10 em 10 com os respectivos botões para mover entre as páginas. No caso da pesquisa de um conjunto de palavras, a página de resultados também apresenta dois botões com as funcionalidades das APIs. A página de administração apresenta duas tabelas com a lista de barrels ativos e top 10 pesquisas.

Endpoints:
GET / : A página inicial do Googol.

POST /sendUrl : Recebe o url introduzido na caixa de texto para indexação e envia-o para a gateway via RMI.

GET /search : Obtém os resultados da pesquisa por um conjunto de palavras e apresenta-os.

GET /urls : Obtém os resultados da pesquisa por url e apresenta-os.

GET /admin : Obtém a lista de barrels ativos e top 10 searches via RMI e apresenta-os.

POST /sendHackerNews : Recebe o conjunto de palavras pesquisado, procura se está em algumas das top stories do hacker news e envia os urls das matched stories para a gateway para indexar.

GET /advice : Obtém um conselho aleatório da api.adviceslip.com e apresenta-o.

De seguida apresentamos na **Figura 2** uma visão geral da arquitetura do relatório.

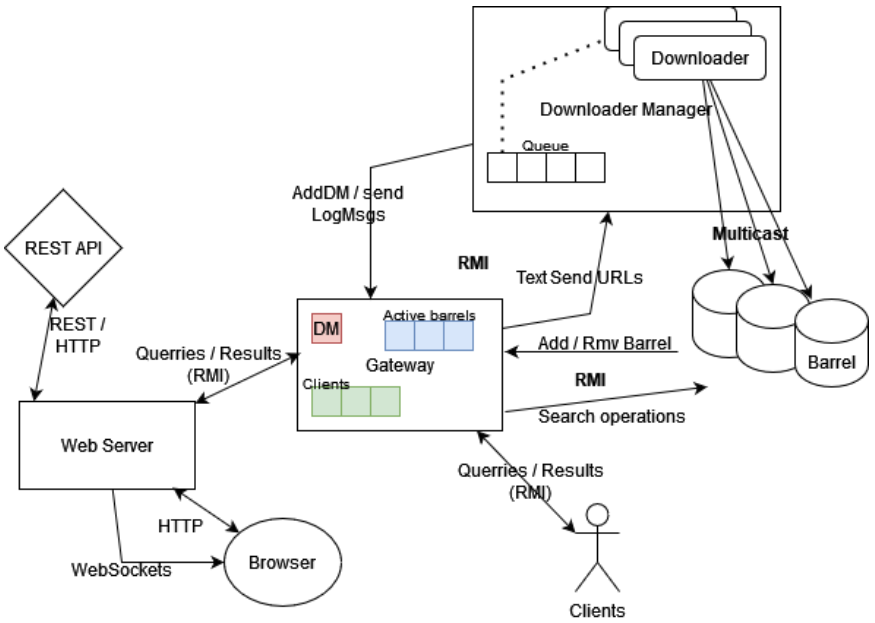

Figura 2: Arquitetura

O código da meta 1 está na pasta server e o da meta 2 na pasta web.

2. Integração de Spring Boot com o servidor RMI

De forma a comunicar com o servidor RMI desenvolvido na meta 1 decidimos dar merge dos projetos e assim correr a meta 1 usando o maven e adicionando o jsoup às dependências.

Assim, o GoogolController efetua uma conexão RMI “global” com a gateway ao iniciar.

Além disso, dentro de todos os endpoints, antes de efetuar qualquer operação, é verificada essa ligação com a gateway e se não tiver ligação, efetua-se uma tentativa da mesma. Desta maneira permite que a web app consiga recuperar a ligação com a gateway.

Os pedidos são feitos como se o GoogolController fosse o Client da meta 1.

3. Integração de WebSockets com Spring Boot e RMI

Para os websockets, no AdminWebSocketController definimos dois destinatários, sendo estes “topic/barrelUpdates” e “topic/searchUpdates”.

Para atualizar a admin page, sempre que um barrel é adicionado ou removido, enviamos com RMI da gateway para todos os clientes (GoogolController) a lista de barrels ativos e estes consequentemente enviam com os websockets para o “topic/barrelUpdates” o que atualiza a admin page em tempo real. No caso das top 10 searches, sempre que um search é efetuado, enviamos a lista de top searches e barrels (visto que o tempo médio de pesquisa dos barrels é atualizado) para todos os clientes e de seguida atualizam a admin page com os websockets.

No caso do cliente clicar no botão para abrir a admin page, os clientes pedem os dados com rmi e não são usados os websockets. Os websockets entram em efeito quando a admin page já está aberta e assim é atualizada em tempo real.

4. Integração de REST WebServices

Os REST WebServices que implementámos foram o API do hacker news tal como pedido e um API que gera um conselho aleatório.

Para a API do hacker news, após uma pesquisa de um conjunto de palavras, está disponível um botão “Hacker News”. Ao clicar é enviado o conjunto de palavras pesquisadas para o /sendHackerNews. Assim, são extraídos os ids das top stories e consequentemente as respectivas stories. Se esse conjunto de palavras estiver presente na story, o url dessa story é enviado para a gateway via RMI para indexação. Após este processo é apresentado ao utilizador um alerta a indicar se houve sucesso nesta operação.

Para a API advice, existe um botão após ter efetuado uma pesquisa de um conjunto de palavras. Este botão envia um pedido GET /advice, que responde com um conselho aleatório, que é apresentado num alerta.

5. Testes

Durante o desenvolvimento da aplicação, conduzimos alguns testes manuais para identificar possíveis erros. Apresentamos os respectivos resultados na **Tabela 1**.

ID	Descrição	PASS / FAIL
1	Indexar um novo URL	PASS
2	URL inválido não causa erros	PASS
3	“search” apresenta o título, citação (caso exista) e links que contenham uma palavra específica	PASS
4	Resultados paginados de 10 a 10	PASS
5	Botões de next e previous no resultados das pesquisas funcionam como esperado	PASS
6	Títulos na página de resultados de pesquisas, quando clicados, redirecionam para o seu site	PASS
7	Quando não há resultados ou acontece algum tipo de erro esse aviso é apresentado na página de results	PASS
8	Pesquisas sem indexação prévia obtêm resultados de indexações efetuadas em execuções anteriores	PASS
9	Pesquisas são feitas por barrels diferentes, estando todas com a mesma informação	PASS
10	Botão de random advice apresenta o resultado esperado num alerta	PASS
11	Botão do hacker news funciona como esperado, embora com um pouco de demora	PASS

12	Quando é indexado um link de uma top storie que contenha uma palavra específica, o DM executa sem crashar ao fim de um tempo	**FAIL**
13	Página de admin atualizada em tempo real	**PASS**
14	Página de Admin apresenta Top 10 organizado	**PASS**
15	Página de Admin apresenta Barrels ativos com respectivos tempos médios de pesquisa	**PASS**
16	Web app ligada antes da gateway recupera a ligação	**PASS**
17	Botão de trocar o tipo de pesquisa funciona como pretendido	**PASS**
18	Página de Sublinks que contenham um link recebido apresenta todos os links por ordem de mais ligações	**PASS**
19	Botões de next e previous na página de Sublinks funcionam como esperado	**PASS**
20	Quando ocorre algum tipo de erro é apresentado o template “error” com erro ocorrido, não havendo qualquer tipo de excepções	**PASS**
21	Web app recupera a ligação com a gateway caso ela crashe	**FAIL**
22	Pesquisas e indexações podem ser feitas por inúmeros clientes e browsers diferentes ao mesmo tempo sem dar problema	**PASS**
25	UI funciona como pretendido não havendo bugs	**PASS**
26	O cliente envia os pedidos e recebe os resultados com sucesso	**PASS**