UNIVERSIDADE D
COIMBRA

Sistemas Distribuídos

Googol - Motor de pesquisa de páginas Web

Ricardo Quintela Martins Santos Rosa - 2020220508
João Rafael do Rosário Henriques - 2020216586
André Filipe Costa Colaço - 2020220301

17 de maio de 2023

Índice

1 Introdução

Este projeto é um sistema distribuído que tem como objetivo criar um motor de pesquisa semelhante aos serviços oferecidos pelos principais motores de busca, como o Google, o Bing e o DuckDuckGo. O sistema permite que os usuários pesquisem páginas da Web com base em palavras-chave e obtenham informações relevantes sobre as páginas que contenham essas palavras-chave entre outras funcionalidades.

Para a construção deste serviço, foi utilizado um trabalho realizado anteriormente. Esse trabalho contém toda a lógica de pesquisa nas bases de dados, a indexação de URLs e o download de informação proveniente desses URLs. Todo esse algoritmo encontra-se explicado no relatório "Googol - Motor de pesquisa de páginas Web"entregue anteriormente.

Para a implementação de uma API que permitisse os pedidos por parte do Web Server, e também que contactasse o Servidor RMI para pedir a informação de forma a enviá-la para o web server, foi utilizado o Spring Boot.

Todas as funcionalidades do projeto anterior foram utilizadas e funcionam da mesma maneira. É possível pesquisar por palavras, indexar novos URLs à fila para passarem por um processo de download e a informação obtida ser guardada, registo e login de utilizadores, procurar por URLs relacionados, apenas no caso em que o utilizador está logado, e ver uma página de administração que contém todas as informações do sistema, como quais os barrels e os downloaders conectados e também as pesquisas mais frequentes.

Foi implementada ainda uma forma de comunicar com o HackerNews[1], de forma a adicionar informação às bases de dados acedidas através do Servidor RMI. Existe assim a possibilidade de pesquisar por TopStories de um determinado autor e indexar todos os seus Urls à fila do servidor RMI, ou ainda pesquisar por Topstories que contenham determinadas palavras e proceder à mesma ação para os seus Urls.

Do site do HackerNews[1] são apenas extaídos os ids das 500 Topstories e, depois, informação de cada uma dessas stories individualmente.

2 Arquitetura do Software

O sistema está construído de uma maneira lógica, ou seja, a informação é guardada em diferentes Models específicos para cada caso, cada endpoint tem a sua funcionalidade e, de entre todas as funcionalidades, a ligação ao HackerNews[1] é feita com threads separadas.

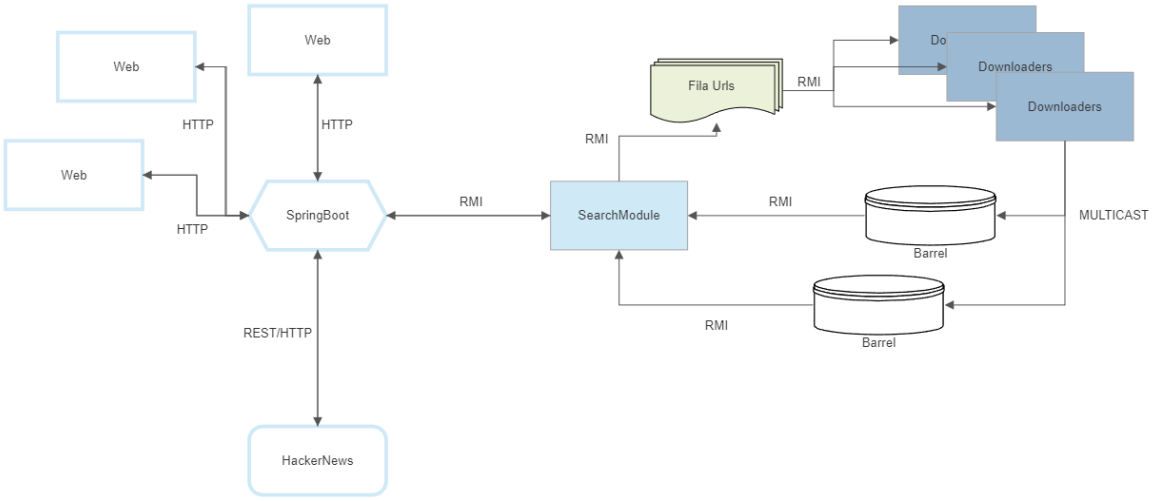

Figura 1: Diagrama de componentes

A figura 1 representa, de uma forma muito resumida, a forma de funcionamento desta aplicação.

2.1 Models

Para uma melhor organização da informação, foram criados alguns models. O envio de resultados contém informação diferente de uma simples confirmação ou mensagem de erro, por exemplo. Por essa razão, existem três Models, *Results*, *HackerNewsRecordItem* e *Popup*.

Num objeto *Results* são guardados 3 atributos, url, título e texto, que são necessários para apresentar ao utilizador após a sua pesquisa. Desta maneira, o tratamento da informação no *HTML* da página ficará mais simples.

Ao receber informação do HackerNews, diversos items são retornados num formato *JSON*, que não são tratados em *Results*. Apesar de não ser necessária toda a informação, o *HackerNewsRecordItem* guarda tudo o que é recebido em atributos diferentes. Desta forma, usar determinadas partes da informação retornada como o autor das TopStories ou o Url da mesma fica simples.

Por último, um objeto *Popup* serve para enviar a informação de erro ou de sucesso para o utilizador. Neste caso apenas é enviada uma string o que será muito mais otimizado do que mandar um objeto *Results* apenas com texto por exemplo.

2.2 Controller

Existe um controller na aplicação que contém todos os métodos necessários ao bom funcionamento de cada uma das funcionalidades.

Inicialmente, ao correr o Spring Boot, é executado um método *@Autowire*, ou seja, é executado sempre no início. Através dele, são lidas as informações de configuração do RMI e é criado um objeto que permite chamar as funções do Serviço RMI.

Para cada endpoint, existe um método que usa o objeto descrito anteriormente, e chama a função que correspondente no Servidor RMI. A informação é retornada para o respetivo metodo que a trata e, se necessário, coloca numa lista de objetos do tipo de um Model e retorna o template correto.

Determinados endpoints, esperam que seja feita uma conexão ao HackerNews[1]. Nesses casos, como o objetivo é sempre indexar URLs na fila do Servidor RMI, o processo é executado sem que o utilizador perceba. O método retorna imediatamente o utilizador para uma página específica e deixa a lógica da indexação a executar numa thread separada.

2.3 Threads

Foram criadas algumas threads para as funções que demoram mais tempo a executar serem tratadas sem que o utilizador tenha de ficar à espera.

A sua criação passa por colocar a anotação *@Async*[2] em cima da função. Deste modo, estamos a definir que a sua execução será feita de forma assíncrona, ou seja, quem chama a função não tem de esperar que ela termine o seu trabalho para prosseguir.

Assim, sempre que for chamada é criada uma thread de forma automática e o utilizador é redirecionado para a respetiva página enquanto a funcionalidade continua a ser executada o tempo que precisar.

2.4 Sockets

Os WebSockets são uma implementação específica de sockets voltada para a comunicação em tempo real através da web. Eles usam uma conexão persistente entre o cliente e o servidor, permitindo a transmissão bidirecional de dados. Enquanto os sockets tradicionais são usados para comunicação em redes locais ou na internet, os websockets são projetados especificamente para comunicação web, sendo suportados por navegadores e servidores web através do WebSocket Protocol.

A informação da administração é passada por WebSockets o que permite que sempre que existir uma alteração a página seja carregada logo com informação atualizada.

3 Integração

A implementação deste serviço necessita de um Servidor RMI que já estava implementado inicialmente. É necessário fazer a integração dos dois lados para que tudo funcione corretamente.

3.1 Servidor RMI

O *Search Module*, ilustrado na figura 1, é o Servidor RMI onde todas as funções necessárias ao bom funcionamento da aplicação estão codificadas. Para lhes aceder é necessário estabelecer a ligação o mais cedo possível.

A conexão deve ser feita apenas uma vez, ao inicializar o Spring Boot. Devido a isso, foi construído um método que faz essa conexão. A informação necessária, endereço, porto e endpoint, está escrita no ficheiro de configurações.

O método contém uma anotação, *@Autowire*, para executar sempre que a aplicação for inicializada. Desta maneira apenas é aberto o socket uma vez por ligação.

3.2 Serviço REST

A conexão ao HackerNews[1], como apresentado na figura 1, é feita através de pedidos REST/HTTP. Esta troca de informações é diferente da utilizada na secção 3.1.

Apesar disso, as funcionalidades desenvolvidas obrigam a que as duas funcionem em simultâneo pois o objetivo de receber informação do HackerNews[1] é enviar os urls das TopStories para a fila do Servidor RMI.

Usando a classe RestTemplate, é possível, apenas com o url e com o tipo de dados, obter a informação desejada. É desta maneira que o Spring Boot recebe a informação neste projeto.

Devido ao elevado volume de dados, as funções que tratam de tudo o que está ligado ao HackerNews[1] são executadas por threads em separado.

3.3 WebSockets

A atualização de informação em tempo real já teve diversos métodos para ser feita mas a maioria esbarrava no mesmo problema, a página tinha de ser atualizada constantemente para uma nova atualização aparecer.

Os WebSockets[4] vieram revolucionar um pouco essa forma de ver as coisas pois permitem comunicação bidirecional, latência reduzida, eficiência no tráfego de rede, notificações em tempo real, maior escalabilidade e integração fácil.

Ao incluir WebSockets no projeto, foram escolhidos alguns urls para os quais a informação é enviada e recebida. A informação que vem do Servidor RMI com a função de adminstração é enviada para o url "/topic/info". A função é chamada de forma regular, com um intervalo de tempo previamente definido através da anotação *@Scheduled*[3].

Para um novo cliente se juntar, subscrever o WebSocket, basta entrar no endpoint da administração. A conexão fica assim estabelecida e só é terminada quando ele sair da página. Este método foi escolhido ao invés da conexão assim que entrasse em qualquer parte do site porque não fazia sentido estar a receber informações caso não estivessemos na página onde estas fossem exibidas.

4 Testes Realizados

A realização de testes é essencial para garantir a qualidade do produto final, identificar possíveis erros ou falhas e corrigi-los atempadamente. Através da seguinte tabela, é possível verificar os testes executados e seus resultados, indicando ainda se passaram ou falharam.

Testes realizados		
-	Descrição	Pass/Fail
Registar novo utilizador com todas as credencias	O cliente consegue registar-se na aplicação com um username e password	Pass
Registar novo utilizador sem password	O cliente não consegue registar-se na aplicação só com username	Pass
Registar novo utilizador sem username	O cliente não consegue registar-se na aplicação só com a password	Pass
Login de um utilizador	O cliente consegue fazer login na aplicação com o seu username e password	Pass
Login de um utilizador sem ter conta	O cliente não consegue fazer login na aplicação sem ter registado	Pass
Indexação de um URL	O cliente pode pedir ao sistema para indexar um novo URL	Pass
Indexação recursiva de URLs	O sistema tem de indexar recursivamente todos os URLs encontrados em páginas previamente visitadas	Pass
Pesquisar por palavras	O cliente pode pesquisar com um conjunto de palavras e o sistema deve mostrar a lista de páginas correspondentes	Pass
Resultados ordenados	O sistema tem de mostrar ao cliente a lista de pesquisas de forma ordenada por ordem de relevância	Pass
Resultados ordenados (usando o *order by* do SQLite)	O sistema tem de mostrar ao cliente a lista de pesquisas de forma ordenada por ordem de relevância	Fail
Paginação das pesquisas	O sistema tem de mostrar ao cliente apenas 10 resultados por página	Pass
Consultar lista de páginas	Os clientes que tenham login efetuado, podem saber, para cada página, todas as ligações que apontem para a mesma	Pass

Testes realizados		
-	Descrição	Pass/Fail
Informações atualizadas	O cliente pode consultar informações como lista de *Downloaders* e *Barrels* ativos e as 10 pesquisas mais comuns	Pass
Informações atualizadas sobre o barrel	Na página de administração, os barrels contém a informação sobre qual a sua partição	Pass
Informações atualizadas após uma pesquisa	A administração atualiza a página em tempo real	Pass
Particionamento do índice	O sistema tem de dividir o indice invertido em duas partes e quando um utilizador faz uma pesquisa deve-se procurar nas duas partes	Pass
Tolerância de falhas dos *Barrels*	Sempre que um *Barrel* falhe, quando voltarem a funcionar têm de recuperar todas as informações que antes tinham	Fail
Proteção dos inputs	Sempre que um input estiver vazio o erro é tratado	Pass

5 Conclusão

Em resumo, este projeto foi uma boa oportunidade para aplicar os conhecimentos teóricos num cenário prático e real. Após a sua realização, podemos destacar a eficiência e a confiabilidade dos sistemas distribuídos quando bem protegidos. Com a implementação de interfaces Web podem-se obter novos problemas que podem levar a diferentes tipos de vulnerabilidades.

Conclui-se que, apesar das vantagens que um sistema distribuído pode trazer, no âmbito da capacidade de processamento ou armazenamento, a sua implementação tem que ser muito rigorosa, uma vez que todas as suas partes devem estar em sintonia e comunicar perfeitamente umas com as outras para o bem do todo.

Além dos benefícios diretos da aplicação dos conhecimentos teóricos, este projeto também nos permitiu desenvolver habilidades importantes, como o trabalho em equipa, a comunicação eficaz e a resolução de problemas. A troca constante de feedback e a colaboração entre os membros da equipa ajudou-nos a melhorar continuamente na realização do projeto. Além disso, a diversidade de opiniões e perspetivas enriqueceu ainda mais a nossa experiência.

Sendo este projeto uma continuação de outro anterior, é possível continuar com o uso desta aplicação e melhorá-la futuramente sobretudo a nível de segurança e performance. Numa perspetiva mais virada para Design, este trabalho pode ter uma continuação para uma reformulação do seu visual.

Referências

[1] Documentation and samples for the official hackernews api. https://github.com/HackerNews/API. 2023-05-17.

[2] How to do *@Async* in spring. https://www.baeldung.com/spring-async. 2023-05-17.

[3] The @scheduled annotation in spring. https://www.baeldung.com/spring-scheduled-tasks. 2023-05-17.

[4] Using websocket to build an interactive web application. https://spring.io/guides/gs/messaging-stomp-websocket. 2023-05-17.