Googol

António Rodrigo Tavares Martins 2021236921

Gustavo Samuel de Alves e Bastos de André e Lima 2020217743

Neste projeto, o nosso objetivo é criar um motor de busca, com funcionalidades semelhantes às do Google.

As funcionalidades deste motor busca são:

1. **Indexação de URLs**: O motor é capaz de indexar novos URLs, e também trata de indexar os URLs encontrados dentro desses sites.
2. **Pesquisa por Termos**: Ele permite pesquisar páginas que contenham um conjunto específico de termos, ordenando os resultados por relevância (ou seja, pelo número de páginas que referenciam cada página).
3. **Indexar URLs das top stories que contenham os termos da pesquisa e stories de um usuário com conta na hackernews específico.**
4. **Consulta de Links**: É possível consultar as páginas que têm um link para uma página específica.
5. **Página de Administração**: Além disso, o projeto inclui uma página de administração com informações importantes sobre os diversos componentes do motor, na qual é ativada em tempo real.
6. **Obter informações sobre o tempo de um determinado local.**

Arquitetura de software

Ao todo são 6 programas: Barrel, Downloader, Gateway, Client, Urlqueue e Web-server.

Barrel

O Barrel atua como um repositório de informações e facilitando a indexação de páginas da web. Este componente opera de forma contínua, recebendo e processando dados provenientes de fontes externas, como o RMI-Gateway e os Downloaders, para garantir a disponibilidade e acessibilidade das informações.

Funcionalidades Principais:
Indexação de URLs: O Barrel recebe URLs provenientes do RMI-Gateway, armazenando e organizando esses links para posterior indexação. Ele mantém um registro das URLs já processadas para evitar duplicações e garantir a eficiência do sistema.

Processamento de Dados: Ao receber informações sobre URLs, o Barrel extrai e armazena dados essenciais, como título, descrição e palavras-chave associadas a cada página da web. Isso é fundamental para a pesquisa de informações posteriormente.

Comunicação com Componentes Externos: O Barrel se comunica com outros componentes do sistema, como os Downloaders, para receber atualizações sobre novas páginas da web a serem indexadas. Além disso, ele interage com o RMI-Gateway para fornecer informações sobre URLs e realizar pesquisas.

Downloader

A função do Downloader é baixar e processar conteúdo de páginas da web em um sistema distribuído. Ele opera continuamente, buscando URLs da Urlqueue, transferindo o conteúdo desses websites e extraindo informações importantes, como título, descrição e palavras-chave.

Para garantir confiabilidade, o downloader lida com diversas situações, como falhas de conexão durante o download, problemas ao obter URLs da fila. Além disso, ele envia via multicast regularmente atualizações e informações sobre os URLs baixados, assegurando que outras partes do sistema estejam informadas sobre o progresso das operações de download.

Durante o download do conteúdo das páginas da web, o downloader utiliza a biblioteca Jsoup para estabelecer conexões e extrair dados essenciais, como título, descrição e palavras encontradas na página. Essas informações são então processadas e encaminhadas para outros elementos do sistema distribuído para análise posterior e indexação.

RMI-Gateway

A classe Gateway desempenha um papel crucial em um sistema distribuído, sendo responsável por facilitar a comunicação com outros componentes do sistema, como os "Barrels", o “RMI-Client” e a “URLQueue”.

Esta classe possui uma variável de instância chamada searches, um HashMap usado para armazenar as pesquisas e o número de vezes que foram pesquisados, bem como as variáveis avgtime que guarda o tempo médio das pesquisas, e num_searches usada para no cálculo de avgtime, permitindo monitorar e registrar o comportamento de pesquisa no sistema. O método construtor inicializa essa variável como um novo HashMap, pronto para armazenar dados assim que uma instância da classe for criada.

O método addLink recebe um URL como entrada e o envia para um servidor remoto através de um socket, realizando uma comunicação de rede vital para adicionar novos URLs ao sistema distribuído. O método linkInfo permite ver informações de um site, a partir dum url,e para além disso permite saber qual a lista de outros URLs que fazem ligação para ele.

Já o método search permite realizar pesquisas por um conjunto de palavras-chave, procurando um objeto remoto no registro RMI e chamando um método search. Esta funcionalidade mostras 10 urls de cada vez, bem como o seu título e descrição;

Por fim, o método getAdminPage retorna os barrels ativos, o tempo médio de pesquisa e as 10 pesquisas mais frequentes realizadas no sistema distribuído.

modular e bem organizada é fundamental para garantir a eficiência e escalabilidade do sistema distribuído.

RMI-Client

O RMI-Client permite interação com o sistema distribuído, oferecendo funcionalidades como adição de URLs, realização de pesquisas, consulta de informações sobre links e visualização de estatísticas de pesquisa. Para isso é fornecido um menu, no qual facilita a escolha pelos usuários.

Para mostrar as páginas de administração em tempo real, foi criada uma thread especial, para ser possível regressar ao menu de opções.

URL-Queue

A URL Queue é responsável por armazenar URLs que precisam ser indexados. Funciona como uma fila, onde os URLs são adicionados pelo Gateway e posteriormente distribuídos pelos Downloaders.

Aqui estão os principais aspectos da URL Queue:

1. Histórico de URLs Indexados: A fila mantém um histórico dos URLs já processados. Isso evita a indexação repetida de URLs idênticos.
2. Aceitamento de URLs:
    - Uma Thread recebe os URLs a serem indexados via TCP..
    - Esses URLs são adicionados à fila.
    - Se um URL começar com “resend”, ele é adicionado à fila independentemente do histórico. Esse tipo de URL é enviado quando um Downloader falha ao processar um URL anteriormente.
3. Envio de URLs:
    - Outra Thread envia os URLs a serem indexados via TCP.
    - Esses URLs são retirados da fila e encaminhados para os Downloaders, que executam o processo de indexação.

Funcionamento da componente multicast

A componente multicast do sistema é responsável pela comunicação entre os downloaders e storage barrels, e também é usada para informar que a página de admin foi atualizada.

A estrutura principal do protocolo é a mensagem composta por um conjunto não ordenado de pares chave-valor, tal como um hashmap.

As informações enviadas não podes conter os seguintes caracteres: (|) “pipe”, (;) “ponto e vírgula” e (\n) “newline”

Ao enviar informações sobre uma url, é usado o seguinte:
“type | url;“
“url | www.website.pt;“
“title | title;“
“description | description;“
“referenced_urls | www.website1.pt www.website2.pt;“
“words | word1 word2 word3;“
A “description“, “referenced_urls“, “title“, “words“ são opcionais

Exemplo de informações de um url:

```
type | url; url | www.uc.pt; title | DEI; description | DEI é fixe; referenced_urls | www.google.com www.facebook.com; words | DEI UC;
```

Para informar que as informações da página de admin foram atualizadas é enviado:

```
type | adminupdated
```

Web-server

Neste projeto, é usado o controlador principal HomeController, que possui diversos métodos para lidar com endpoints diferentes:

- / - Página inicial em que é possível fazer uma pesquisa com uns conjunto de termos

O model usado é usado para as informações meteorológicas da página inicial:

Nome do atributo | Descrição do atributo
“weatherIcon” | Emoji que representa o estado do tempo
”weather” | Estado do tempo
”Temp” | Temperatura atual em graus celsius
”humidity” | Percentagem de humidade
”windSpeed” | Velocidade do vento

- /search - Página onde são mostrados os resultados da pesquisa

Aqui é usado o método search do servidor RMI para obter resultados de uma pesquisa

Parâmetros de url:
.Nome do parâmetro | Descrição do parâmetro
q | string query de pesquisa (opcional)
page | int index dá página (opcional)

O model usado é usado para a obtenção dos resultados das pesquisas, bem como o index da página atual para esconder o botão de ir para a página anterior e a query string para inserir na input box.
.Nome do atributo | Descrição do atributo
”results” | string com o resultados das pesquisas
”q” | query com a string da pesquisa
”page ” | index da página atual

- /linkinfo - Página onde é possível consultar as páginas que têm um link para uma página específica.

Aqui é usado o método linkInfo do servidor RMI para obter informações de uma página

Parâmetros de url:
Nome do parâmetro | Tipo | Descrição do parâmetro
q | string | query com o url a obter informaçoes (opcional)

O model contém apenas uma as informações sobre um determinado url.
Nome do atributo | Descrição do atributo
”results” | string com as informações sobre um determinado url, como título, descrição e páginas que apontam para esse url.

- /addlink - Página onde é possível adicionar um url para indexar.

Aqui é usado o método addLink do servidor RMI para indexar uma página com um determinado url

Parâmetros de url:
.Nome do parâmetro | Tipo | Descrição do parâmetro
q | string | query de pesquisa (opcional)

O model é usado para a informar se foi indexado com sucesso
Nome do atributo | Descrição do atributo
“results” | String com resposta obtida do gateway, a informar se foi indexado com sucesso

- /admin - Página de administração com informações importantes sobre os diversos componentes do motor, na qual é ativada em tempo real

Aqui é usado o método getAdminPage do servidor RMI para obter as informações da página de administração. A primeira vez que esta é chamada é ao carregar a página, com o argumento wait = false, para o usuário ter já noção das informações atuais da página.

Posteriormente estas informações serão recebidas via socket, em que esta é chamada com com o argumento wait = true, por parte do servidor.

O model contém informações relevantes como tempo médio de pesquisa, barrels ativos e top 10 pesquisas
Nome do atributo | Descrição do atributo
“results” | String com barrels ativos, top 10 pesquisas e tempo médio de pesquisa nos barrels

- /indexhacker - Permite indexar as principais notícias HackerNews com termos especificos

Aqui é usado o método addLink do servidor RMI para indexar as principais notícias HackerNews.

Parâmetros de url:
Nome do parâmetro | Tipo | Descrição do parâmetro
q | string | query de pesquisa (opcional)

O model é para mostrar quais os urls que serão indexados
Nome do atributo | Descrição do atributo
“results” | String com os urls que serão indexados

- /indexhackerbyuser - Permite indexar páginas de um determinado usuário HackerNews

Aqui é usado o método addLink do servidor RMI para indexar as páginas submetidas por um determinado usuário.

Parâmetros de url:
Nome do parâmetro | Tipo | Descrição do parâmetro
id | string | username para indexar as suas páginas submetidas (opcional)

O model é para mostrar quais os urls que serão indexados
Nome do atributo | Descrição do atributo
“results” | String com os urls que serão indexados

- /weather - Permite obter informações de meteorologia de um determinado local

Parâmetros de url:
Nome do parâmetro | Tipo | Descrição do parâmetro
city | string | cidade com meteorologia a pesquisar (opcional)
lat | double | latitude com meteorologia a pesquisar (opcional)
lon | double | longitude com meteorologia a pesquisar (opcional)

O model é para mostrar quais os urls que serão indexados
Nome do atributo | Descrição do atributo
“results” | String com os urls que serão indexados

Integração de WebSockets com Spring Boot e RMI

A classe Main utiliza o SimpMessagingTemplate do Spring para enviar mensagens para o tópico "/topic/messages" atualizando as informações da página de administração, obtidas as informações via rmi usando o método getAdminPage com argumento wait = true, e assim que obter uma atualização, envia para os clientes que estão subscritos ao tópico '/messages'.

Do lado do cliente é criado um socket usando SockJS conectando ao endpoint '/s' do servidor e depois é subscrito ao tópico 'messages', e ele aí recebe as informações, Sempre que a conexão for perdida, tentara em novamente em 10 segundos.

REST WebServices

Para interagir com as APIs, foram usadas a biblioteca RestTemplate para fazer chamadas às APIs.

O controlador interage os 2 seguintes serviços:
WeatherService: Fornece dados meteorológicos da API OpenWeatherMap.
HackerNewsService: Fornece as principais notícias da API Hacker News.

Para WeatherService, foi criado uma classe modelo:

- WeatherResponse, para mapear a resposta das meteorológicas obtidas da API do OpenWeatherMap

Operações de obtenção da meteorologia realizadas:
Método GET
https://api.openweathermap.org/data/2.5/weather?lat={lat}&lon={lon}&appid={API key}
Parâmetros de url:
Nome do parâmetro | Descrição do parâmetro
lat | latitude com valores decimais entre (-90; 90) da qual é pretendida obter informação sobre a meteorologia (obrigatório)
log | longitude com valores decimais entre (-180; 180) da qual é pretendida obter informação sobre a meteorologia (obrigatório)
API key | API Key usada para autenticar solicitações à API do OpenWeather(obrigatório)

Método GET
https://api.openweathermap.org/data/2.5/weather?q={city name}&appid={API key}
Parâmetros de url:
Nome do parâmetro | Descrição do parâmetro
city name | Cidade na qual é pretendida obter informação sobre a meteorologia (obrigatório)
API key | API Key usada para autenticar solicitações à API do OpenWeather(obrigatório)

Das informações obtidas, as usadas são a temperatura (temp que vem é graus kelvin), o tempo (weather), a percentagem de umidade (humidity em percentagem), a velocidade do tempo (windSpeed em km/h) e a cidade.

Para HackerNewsService foram criadas as classes modelos:

- StoryResponse para mapear os storys: StoryResponse
- UserResponse para mapear usuários: UserResponse

Operações realizadas:
Obter top storys:
Método GET
https://hacker-news.firebaseio.com/v0/topstories.json

Obter informações de um story:
Método GET
https://hacker-news.firebaseio.com/v0/item/{itemid}.json
Parâmetros de url:
Nome do parâmetro | Descrição do parâmetro
itemid | id de story a obter informações de stories, comments, jobs

Obter informações de um utilizador:
Método GET
https://hacker-news.firebaseio.com/v0/user/{userid}.json
Parâmetros de url:
Nome do parâmetro | Descrição do parâmetro
userid | id do utilizador a obter comentarios e submissões de storys

A classe HackerNewsService é um serviço no Spring Boot responsável por interagir com a API do Hacker News. ela possui os seguintes métodos:

- getTopnews retorna uma lista com as topstorys em que contenham termos específicos. Na primeira chamada à api, é obtido os topstorys, nas seguintes são feitas uma chamada para cada um dos topstorys, para obter o seu url. posteriormente é verificado se a story contém o conjunto de termos. Posteriormente é retornado os urls dos storys para que possam ser indexados.

- getIdnews retorna uma lista com as storys de um determinado utilizador. Na primeira chamada à api, é obtido os todos os storys submetidos, nas seguintes são feitas uma chamada para cada um dos topstorys, para obter o seu url. Posteriormente é retornado os urls dos storys para que possam ser indexados.

Testes de software (tabela: descrição e pass/fail de cada teste)

Exceder o número máximo de obtenções de informações sobre meteorologia	pass
Para o cliente que estiver a receber informações da página de administração, encerrar o servidor para quebrar a conexão entre cliente e o servidor, e voltar a	pass

abrir para verificar se conecta novamente com sucesso	
Verificar se é indexado as páginas do top hacker news	pass
Verificar se é indexados as páginas submetidas por um user são indexadas	pass
Testar páginas com índices negativos e verificar se não é apresentado erro	pass
Testar a recepção de mensagens em vários clientes simultaneamente na página de administração	pass
Introduzir um url invalido para obter informações de uma página	pass
Tentar indexar um url já indexado	pass
Desligar um barrel e forçar que faça uma pesquisa em noutro barrel	pass

Distribuição de tarefas pelos elementos do grupo

O projeto foi realizado de forma colaborativa através de chamadas no Discord, onde partilhamos o ecrã para facilitar a comunicação e o acompanhamento em tempo real. Além disso, mantivemos um servidor em funcionamento para monitorizar o desenvolvimento do site e utilizámos o GitHub para gestão de código e controlo de versões.