

# **Googol: Motor de pesquisa de páginas Web**

## **Sistemas Distribuídos**

17 de maio de 2023

---

Gonçalo Senra 2020213750 PL2  
uc2020213750@student.uc.pt

João Coelho 2020235901 PL2  
uc2020235901@student.uc.pt

---

Raul André Brajczewski Barbosa  
Hugo Dinis Pereirinha da Silva Amaro

---

# Índice

<b>INTRODUÇÃO.....</b>	<b>3</b>
<b>ARQUITETURA DE SOFTWARE .....</b>	<b>3</b>
<i>Indexar novo URL.....</i>	<i>4</i>
<i>Pesquisar páginas que contenham um conjunto de termos.....</i>	<i>5</i>
<i>Pesquisar por links que apontem para um link.....</i>	<i>6</i>
<i>Página de administração .....</i>	<i>6</i>
<i>HackerNews .....</i>	<i>7</i>
<b>INTEGRAÇÃO COM O SERVIÇO REST.....</b>	<b>8</b>
<i>INTEGRAÇÃO COM A API HACKER NEWS.....</i>	<i>8</i>
<b>INTERAÇÃO ENTRE O SPRING BOOT E O SERVIDOR RMI .....</b>	<b>9</b>
<b>TESTES REALIZADOS À PLATAFORMA.....</b>	<b>10</b>
<b>CONCLUSÃO .....</b>	<b>12</b>
<b>BIBLIOGRAFIA.....</b>	<b>13</b>

# Introdução

Nesta fase final do trabalho foi-nos pedido para desenvolver uma aplicação web de modo a fornecer um serviço, serviço esse que irá utilizar como backend todos os programas desenvolvidos na meta anterior. Logo no decorrer deste relatório, iremos abordar em detalhe a arquitetura do software desenvolvido, a utilização de serviços como Spring Boot, webSockets e REST e também a interação destes com o trabalho desenvolvido na meta anterior.

## Arquitetura de software

Tal como requerido pelos professores, para implementar todas as funcionalidades, utilizamos a arquitetura MVC (Model-View-Controller), que é baseada na premissa de que os controladores mediam a informação entre os modelos (manipulação e validação de dados) e as views (interação com o utilizador). Deste modo, depois de discutirmos qual seria a melhor organização, decidimos que seria uma boa abordagem se dividíssemos os controladores por funcionalidade.

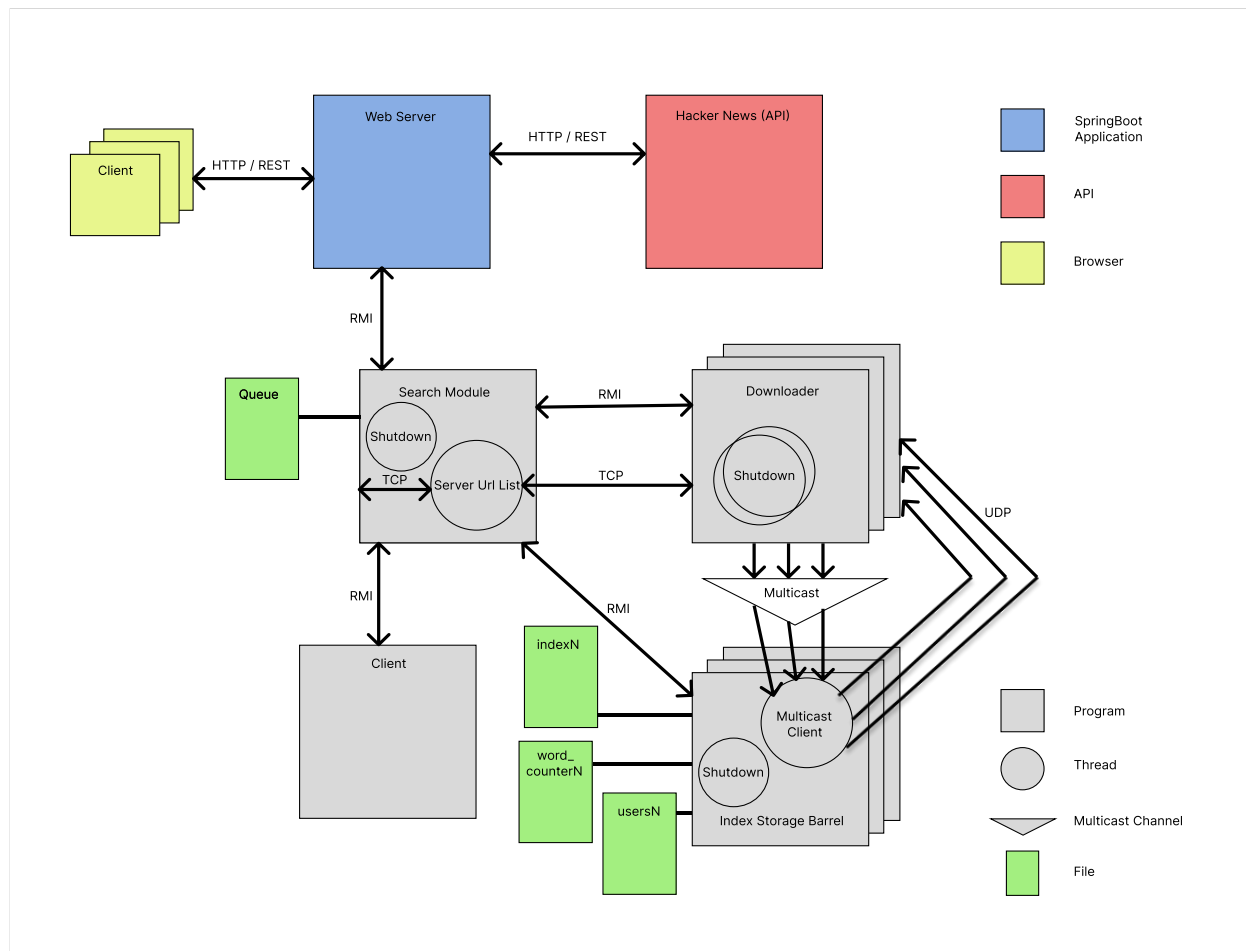
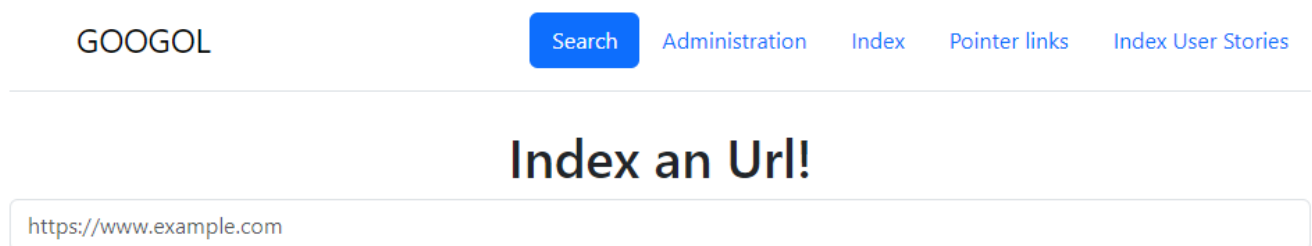


Figura 1 - Arquitetura do software

# Indexar novo URL

Esta funcionalidade está assente no controller `IndexController.java`, este tem dois endpoints:

- **@GetMapping("/index")** – este mapping apresenta ao utilizador o template `index.html`, e adiciona um atributo “url”, que por sua vez vai guardar o url que o utilizador digitar, para enviar para indexação. O template `index.html` tem uma tag `<form>` que, quando um utilizador submete um link redireciona para o endpoint “`index/send_url`”.
- **@PostMapping("/index/send\_url")** – este mapping recebe por parâmetro o url do utilizador, e chama uma função para indexar, de uma instância do objeto `WebServerRMI` previamente criada na classe principal do programa, cuja funcionalidade é servir os controladores com as funcionalidades da Meta 1, utilizando a comunicação RMI indicada no diagrama da arquitetura (**figura 1**).



GOOGOL

Search Administration Index Pointer links Index User Stories

## Index an Url!

*Figura 2 - Interface para indexar um URL*

## Pesquisar páginas que contenham um conjunto de termos

Esta funcionalidade foi implementada no SearchController.java, que é um controlador que contém o seguinte endpoint:

- **@GetMapping("/search")** – este endpoint recebe por parâmetro os tokens de uma pesquisa e o número da página da mesma, com esses dados a função invoca um método do objeto WebServerRMI para pedir ao Search Module o resultado de uma pesquisa, e conforme o resultado obtido, o template “search.html”, mostra os links encontrados. Para além disto, existe um botão que quando pressionado envia os tokens para o endpoint “/HackerNews” (explicado mais à frente).

GOOGOL [Search](#) [Administration](#) [Index](#) [Pointer links](#) [Index User Stories](#)

---

Search!

[Search](#) [Index from Hacker News](#)

<https://www.neucards.com/#products>

**welcome to neucards**  
People move, switch jobs, change phone numbers and over time traditional contact lists become hopelessly out of date. With nεucards, the digital contact cards you receive update automatically, so you always have the latest information.  
[Visit](#)

<https://www.neucards.com/>

**welcome to neucards**  
People move, switch jobs, change phone numbers and over time traditional contact lists become hopelessly out of date. With nεucards, the digital contact cards you receive update automatically, so you always have the latest information.  
[Visit](#)

*Figura 3 - Interface para procurar palavra(s)*

# Pesquisar por links que apontem para um link

Esta funcionalidade é suportada pelo `PointersController.java`, com o seguinte endpoint:

- **@GetMapping("/pointers")** – este endpoint recebe um link por parâmetro e envia-o para o `SearchModule`, com o intuito de receber ok links que apontam para esse link. O template responsável pela interface é o `"pointers.html"`.

GOOGOL  [Administration](#) [Index](#) [Pointer links](#) [Index User Stories](#)

---

## Search for pointer links!

<https://www.abola.pt/nnh/2023-04-24/liga-o-que-falta-jogar-a-benfica-fc-porto-sc-braga-e-sporting/979950>

<https://www.abola.pt/nnh/noticias/ver/985197>

**A BOLA - Grimaldo apontado à Premier League (Benfica)**  
O Fulham, orientado por Marco Silva, estará atento à situação de Álex Grimaldo no Benfica.

<https://www.abola.pt/nnh/2023-04-26/benfica-avancado-nigeriano-apontado-a-aguia/985263>

**A BOLA - Avançado nigeriano apontado à águia (Benfica)**  
Gift Orban, ponta de lança nigeriano de 20 anos dos belgas do Gent, que esta época tem 14 golos em 16 jogos, foi ontem associado pelo Bola na Rede a um potencial interesse do Benfica.

*Figura 4 - Interface para procurar links que apontem para um link*

## Página de administração

O `WebServer` é constantemente atualizado, quando alguma informação relevante é alterada (nº de barrels ativos, nº de downloaders ativos e palavras mais pesquisadas), contudo a mesma não é atualizada em tempo real, tal como era pedido no enunciado, já que a implementação dos `Web Sockets` não foi bem-sucedida. A página de administração foi mapeada para o `AdministrationController.java`, o mesmo contém o seguinte endpoint:

- **@GetMapping("/admin")** – este endpoint é responsável por enviar a informação relevante para o template “admin.html”.

GOOGOL

SearchAdministrationIndexPointer linksIndex User Stories

---

Administration page!

Barrels

ID: 0 IP: 172.24.208.1

Downloaders

Nothing to see here 🔍

Top Searches

benfica (22 search(es))
otamendi (6 search(es))
ccecewce (1 search(es))
people (1 search(es))
grimaldo (1 search(es))

*Figura 5 - Interface da página de administração*

## HackerNews

No que diz respeito à interação do WebServer com a API HackerNews, todas as funcionalidades pedidas foram implementadas com sucesso e detalhadamente explicadas no capítulo [Interação entre o Spring Boot e o Servidor RMI](#). Esta funcionalidade foi implementada nos endpoints:

- **@PostMapping("/HackerNews")** – responsável por procurar todas as stories que contenham os tokens introduzidos, e caso tenham link enviá-lo para indexar.
- **@GetMapping("/HackerNews/users")** – este mapping recebe um nome de utilizador, e se o mesmo for válido, todas as stories submetidas por ele são enviadas para indexar.



## Index User Stories (HackerNews)!

User Stories sent for index!

*Figura 6 - Interface indexar User Stories (HackerNews)*

## Integração com o serviço REST

### INTEGRAÇÃO COM A API HACKER NEWS

De acordo com o enunciado, um utilizador do nosso serviço teria de ser capaz de, durante uma pesquisa, indexar todos os **links das top stories** do **HackerNews**, cujo texto contivesse a(s) palavra(s) inseridas pelo mesmo. Para esta funcionalidade, e após ler atentamente a documentação da API, decidimos que a melhor estratégia seria estabelecer uma conexão com o endpoint “<https://hacker-news.firebaseio.com/v0/topstories.json?print=pretty>” e através do mesmo obtivemos um **JSONArray** com todos os **Ids das top stories** (500 stories), de seguida, por cada Id contido nesse array (por questões de eficiência decidimos apenas contar com as primeiras 100 stories), estabelecemos uma conexão com o endpoint “<https://hacker-news.firebaseio.com/v0/item/{ID}.json?print=pretty>” ,e retiramos um JSON da resposta, caso esse JSON tivesse um url, então esse url seria enviado para o **Search Module** para ser indexado, caso contrário continuava a iteração. (*Figura 3*)



Para além desta funcionalidade, um utilizador do Googol teria a possibilidade de **indexar todas as stories de um utilizador**, nesse caso seria estabelecida uma ligação com o endpoint “<https://hacker-news.firebaseio.com/v0/user/{USER}.json?print=pretty>”, caso o utilizador não existisse, a resposta devolvida pela API seria “**null**”, e consequentemente, uma mensagem de erro seria enviada para a interface, por outro lado, se o utilizador fosse válido, seria devolvido um JSON com toda a informação desse utilizador, inclusive com todas as stories submetidas até ao momento.

Após retirada a informação das stories submetidas, mais uma vez, iteramos os Ids das mesmas e retiramos o link da resposta a este endpoint “<https://hacker-news.firebaseio.com/v0/item/{ID}.json?print=pretty>”, com isto, bastava que a story tivesse url para ser enviada para indexação. (*Figura 6*)

## Interação entre o Spring Boot e o Servidor RMI

Uma parte crucial do bom funcionamento de toda a aplicação foi assegurar uma conexão entre o nosso **WebServer** e todas as aplicações desenvolvidas na meta anterior, como o **SearchModule**, **Donwloaders** e **Barrels**, pois sem este tipo de conexão seria virtualmente impossível satisfazer os pedidos dos clientes. Para isso, sempre que a nossa aplicação web for iniciada será criado um registo chamado “**WebServer**”, registo esse que estará associado a uma interface do **SearchModule**. Por outro lado, no **SearchModule** também iremos criar uma interface do **WebServer**.

Posto isto, já podemos explicar os principais usos desta interface no nosso **WebServer**:

1. Quando o cliente quiser realizar uma pesquisa, o mesmo precisa de navegar até ao endpoint “**/search**”, uma vez neste endpoint, caso seja detetado que o cliente pressionou o botão “**Search**” após ter escrito algo na caixa de texto, o controller “**SearchController**” irá chamar o método “**Search**”, método esse que irá dar uso à interface criada anteriormente e executar o método “**SearchLinks**”, sendo que este método será executado no **SearchModule**. Uma vez devolvida a informação, o cliente será automaticamente redirecionado para uma página onde serão apresentados os resultados. (Nota: explicação do método “**SearchLinks**” no relatório da meta 1 do trabalho prático);
2. Caso o cliente queira indexar um link, o mesmo necessita de navegar até ao endpoint “**/index**”. Uma vez neste endpoint, caso seja detetado que o cliente pressionou o botão, será chamado o método “**IndexLink**” que irá executar no **SearchModule** o método “**IndexUrl**”. De Seguida será

mostrada uma mensagem no ecrã para informar o cliente do sucesso, ou falha da operação. (Nota: explicação do método “**IndexUrl**” no relatório da meta 1 do trabalho prático);

3. De seguida, caso o cliente queira consultar as páginas com ligação para uma página específica, se se encontrar no endpoint “**/pointers**” e pressione o botão após escrever um url na caixa de texto, será executado o método “**SearchPointers**” que irá utilizar o método “**SearchPointers**”, mais uma vez este último método será executado no SearchModule. (Nota: explicação do método “**SearchPointers**” no relatório da meta 1 do trabalho prático);
4. Caso o cliente queira realizar alguma operação relacionada com o serviço “**Hacker News**”, os métodos usados irão ambos executar o método “**IndexLink**”, que mais uma vez, irá executar o método “**IndexUrl**” no SearchModule. (Nota: explicação do método “**IndexUrl**” no relatório da meta 1 do trabalho prático; Explicação mais detalhada das operações relacionadas a este serviço no capítulo [Arquitetura de Software](#));
5. Por fim, sempre que forem executados métodos de criação de barris, downloaders ou pesquisas no SearchModule, irão ser executados métodos que vão atualizar a informação necessária para a página de administração. Esses métodos irão usar a interface do **WebServer** presente no **SearchModule** para atualizar as listas de informação presentes no webServer. Deste modo quando o cliente utilizar o endpoint “**/admin**” apenas iremos utilizar os métodos “**getInfoBarrels**”, “**getInfoDownloaders**” e “**getTopSearches**”, executados no webServer.

## Testes realizados à plataforma

1. O cliente consegue dirigir-se ao endpoint “**/search**”, escrever uma palavra na caixa de texto e clicar no botão “**Search**” para realizar uma pesquisa.

**Resultado:** Aprovado – É apresentado no ecrã o resultado da pesquisa.

2. O cliente consegue dirigir-se ao endpoint “**/index**”, escrever um url na caixa de texto e clicar no enter para indexar um url.

**Resultado:** Aprovado – É apresentada no ecrã uma mensagem de sucesso ou falha.

3. Quando é efetuada o pedido de indexação de um link, o link é tratado pelos **downloaders** e guardado nos **storage barrels**.

**Resultado:** Aprovado.

4. O cliente consegue dirigir-se ao endpoint “**/pointers**”, escrever um url na caixa de texto e clicar no enter para obter todas as páginas com ligação para uma página específica.

**Resultado:** Aprovado – São apresentados no ecrã todas páginas com ligação para a página específica.

5. O cliente consegue dirigir-se ao endpoint “**/admin**” para obter a informação sobre todos os **barrels** e **downloaders** assim como a lista das palavras mais pesquisadas.

**Resultado:** Aprovado.

6. O cliente consegue movimentar-se facilmente pela aplicação web com recurso a vários botões.

**Resultado:** Aprovado.

7. O cliente consegue voltar a página principal ao clicar no botão “**Googol**”.

**Resultado:** Aprovado.

8. Se forem detetadas alterações nos **barris** ou **donwoladers** ativos, assim como as palavras mais pesquisadas, a informação presente no endpoint “**/admin**” é atualizada ao pressionar f5.

**Resultado:** Aprovado.

9. Se forem detetadas alterações nos **barris** ou **donwoladers** ativos, assim como as palavras mais pesquisadas, a informação presente no endpoint “**/admin**” é atualizada automaticamente em tempo real.

**Resultado:** Reprovado – é obrigatório o cliente pressionar f5 ou atualizar a página para as alterações serem apresentadas.

10. Caso o cliente queira indexar todos os urls que contenham uma palavra específica nas “top stories” do Hacker News, escrever a palavra requerida na caixa de texto e clicar no botão “**Index from Hacker News**”.

**Resultado:** Aprovado – É apresentada no ecrã uma mensagem de sucesso ou falha.

11. Caso o cliente queira indexar todas as stories de um dado utilizador do Hacker News, basta o mesmo basta o mesmo dirigir-se ao endpoint “**/HackerNews/users**” escrever o username do utilizador requerido na caixa de texto e clicar no enter.

**Resultado:** Aprovado – É apresentada no ecrã uma mensagem de sucesso ou falha.

12. Caso um cliente execute uma pesquisa, os resultados dessa pesquisa apenas serão mostrados a esse cliente, ou seja, caso um novo cliente se ligue ao Googol os resultados da pesquisa anterior não serão mostrados.

**Resultado:** Aprovado.

13. Caso um cliente execute uma pesquisa, abra outro endpoint do Googol e volte ao endpoint “**/search**”, os resultados da pesquisa anterior não serão mostrados.

**Resultado:** Aprovado.

## Conclusão

Agora que todos os aspetos do trabalho foram expostos, apesar de não termos conseguido concluir por completo a realização de uma página de administração atualizada em tempo real sem a necessidade de atualizar a página, achámos ter atingido a grande maioria dos objetivos propostos no enunciado. No entanto, todas as funcionalidades necessárias foram implementadas e funcionam sem qualquer problema, tendo em conta os testes executados e enunciados no capítulo anterior.

Para finalizar, cremos ter compreendido a importância e necessidade da segunda meta deste trabalho, pois em aplicações destas, criadas para fornecer um serviço a clientes, não faria sentido desenvolvê-la apenas num ambiente local. De notar também que acreditamos que os serviços estudados e necessários para desenvolver esta meta poderão ser de extrema importância para o futuro, sendo que a sua compreensão apenas foi alcançada pelos temas abordados durante as aulas e também pelo código fornecido pelos docentes.

# Bibliografia

- Ficheiros fornecidos pelos docentes [10/05/2023];
- [Spring Initializr](#)
- <https://www.baeldung.com/spring-bean-scopes>
- [Maven Repository: Search/Browse/Explore \(mvnrepository.com\)](https://maven.apache.org/search/browse/explore.html)
- <https://getbootstrap.com>
- <https://github.com/HackerNews/API>
- <https://www.w3schools.com/>
- <https://docs.spring.io/spring-boot/docs/3.0.6/reference/htmlsingle/#io>