Uma imagem com Simetria, file, origami, arte

Descrição gerada automaticamente

|  |
| --- |
| Googol  Motor de pesquisa |
|  |
| 2ª Meta  Da autoria de:  Eduardo Figueiredo 2020213717  Fábio Santos 2020212310 |

**Índice**

[Introdução 2](#_Toc135218670)

[Arquitetura de Software 3](#_Toc135218671)

[WebServer 4](#_Toc135218672)

[Hacker News API 4](#_Toc135218673)

[Integração entre o Spring Boot e o Search Module 5](#_Toc135218674)

[Login 5](#_Toc135218675)

[Indexar um URL 6](#_Toc135218676)

[Pesquisa segundo um conjunto de termos 7](#_Toc135218677)

[Pesquisa de URLs que têm uma ligação para um dado URL 9](#_Toc135218678)

[Integração com a REST API do Hacker News 10](#_Toc135218679)

[Indexar as Top Stories do Hacker News que contenham um conjunto de termos 10](#_Toc135218680)

[Indexar as Stories de um dado utilizador do Hacker News 11](#_Toc135218681)

[Notificações em tempo real 12](#_Toc135218682)

[Alterações do Backend face à Meta 1 14](#_Toc135218683)

[Testes Realizados 15](#_Toc135218684)

[Conclusão 17](#_Toc135218685)

# Introdução

Este projeto tem como objetivo pôr em prática os conhecimentos lecionados na cadeira de Sistemas Distribuídos, criando um motor de busca semelhante ao Google ou Bing. Esta meta teve como propósito criar uma GUI (Graphical User Interface) acessível através do browser. Assim, uma vez tendo desenvolvidos todos os requisitos da primeira meta, desta vez apenas foi necessário adaptar o trabalho de backend feito na primeira meta e dar-lhe um frontend acessível por um site. Este site foi maioritariamente criado com a framework Spring Boot do Java, em conjugação com o Thymeleaf e outras bibliotecas necessárias.

# Arquitetura de Software

A Arquitetura de Software pouco mudou em relação à primeira meta. Como é possível ver através da *Figura 1*, todos os componentes do backend mantém-se iguais, apenas sendo acrescentado o novo componente do frontend: o WebServer. Desta forma, a forma de comunicação entre os componentes do backend foi inalterada enquanto os próprios componentes foram, em alguns casos, ligeiramente modificados de forma a dar suporte às novas funcionalidades do frontend, como iremos explicar numa secção mais adiante.

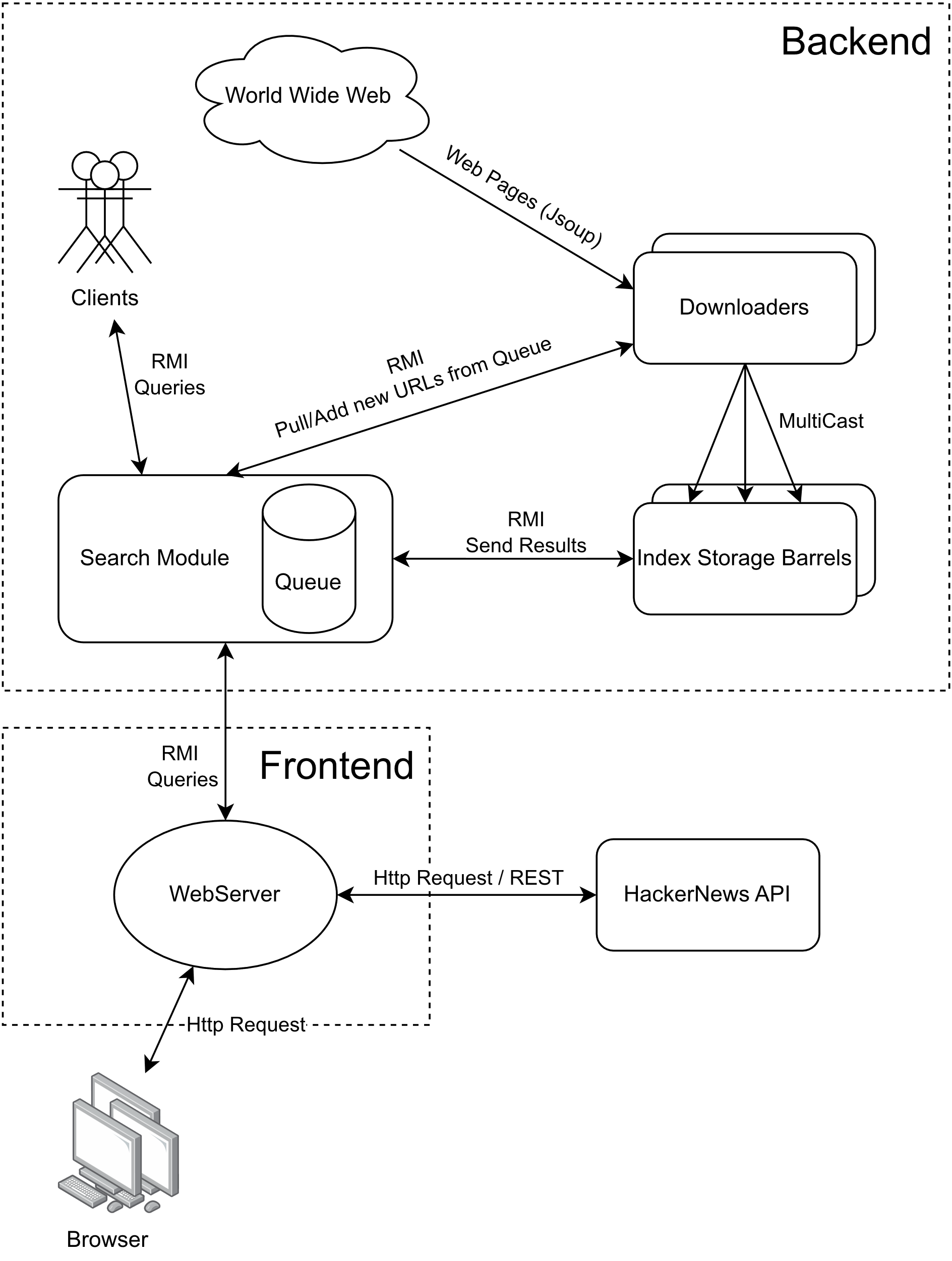


Figura 1

## **WebServer**

O WebServer é o componente responsável por aceder a um Endpoint (URL) facultado pelo utilizador na barra de pesquisa do browser e, se este existir, fazer a comunicação com o Search Module para de seguida mostrar o conteúdo recebido. Depois de receber a informação enviada pelo Search Module, estes componentes comunicam entre si através de RMI (Remote Method Invocation). Uma vez que já existiam as interfaces necessárias para comunicações RMI entre o Search Module e outros componentes do backend, simplesmente reutilizaram-se estas interfaces para fazer a correspondência com o backend. Assim, os pedidos ao servidor que outrora foram possíveis fazer pela aplicação do Cliente no terminal, agora também são possíveis fazer através da Web num site responsivo e apelativo.

O WebServer é constituído apenas por um Controller que faz a gestão de todos os endpoints do sistema. Este Controller (Controller1.java) é capaz de lidar com endpoints errados ou outro tipo de erros redirecionando o utilizador para uma página de erro.

É também usado um WebSocket usado na página de administração de forma a mostrar a informação atualizada em tempo real sem que o utilizador necessite de atualizar a página (mais detalhes sobre a implementação mais a frente).

## **Hacker News API**

A API (Application Programming Interface) do Hacker News é uma componente externa ao Sistema, porém essencial ao funcionamento dele. Isto porque não é uma componente programada por nós, mas sim uma componente usada para extrair informação a injetar nos Index Storage Barrels e consequentemente mostrá-la.

# Integração entre o Spring Boot e o Search Module

De forma a fazer a ponte entre o WebServer do Spring Boot e o Search Module da Meta 1 foi usada a GoogolInterface.java. Assim, é possível invocar funções do Search Module no WebServer de forma remota.

De notar que as imagens foram retiradas da demostração feita em localhost, pelo que os endpoints mostrados começaram com “localhost:8080/”.

## **Login**

Endpoint: <http://localhost:8080/login>

Uma imagem com texto, captura de ecrã, software, Sistema operativo

Descrição gerada automaticamenteO Login é a página onde um utilizador se pode autenticar. Caso um utilizador não tenha conta na aplicação pode sempre clicar no link “Register” para criá-la, como mostra a *Figura 2*.

Figura 2

Este passo é obrigatório antes de aceder a qualquer outra funcionalidade do sistema. No caso de o utilizador tentar aceder ao Endpoint da pesquisa ou ao Endpoint das Top Stories do Hacker News sem que este tenha efetuado o login, o seu acesso é negado, sendo redirecionado para a página de login.

Quando um utilizador efetua o login com sucesso é redirecionado para a página de pesquisa podendo aceder a qualquer uma das próximas funcionalidades através da barra de navegação presente no topo de todas as páginas.

As próximas figuras que representam os Endpoints das funcionalidades desenvolvidas foram acedidas por um utilizador denominado “fabio”.

## **Indexar um URL**

Endpoint: <http://localhost:8080/index>

Uma imagem com texto, captura de ecrã, computador, software

Descrição gerada automaticamenteUm utilizador logado (com o login efetuado) consegue indexar um link a ser acoplado ao sistema. O URL inserido pelo utilizador na caixa de texto apresentada na *Figura 3* viaja por RMI para o Search Module para ser analisado pelo Downloader e guardado no(s) Barrel(s), como explicado no Relatório da Meta 1.

Figura 3

Uma mensagem de sucesso é apresentada se o link tiver sido adicionado à Queue de links por processar, como é mostrado na *Figura 4*.

Uma imagem com texto, captura de ecrã, computador, Sistema operativo

Descrição gerada automaticamente

Figura 4

## **Pesquisa segundo um conjunto de termos**

Endpoint: <http://localhost:8080/search>

Um conjunto de termos pode ser inserido na barra de pesquisa mostrada na *Figura 5* para pesquisar o conjunto de URLs que contenham pelo menos uma dessas palavras.

Uma imagem com texto, captura de ecrã, computador, software

Descrição gerada automaticamente

Figura 5

Uma imagem com texto, captura de ecrã, software, Página web

Descrição gerada automaticamenteO resultado é apresentado numa tabela (*Figura 6*) que contém colunas para o URL em si, o título da página e uma breve descrição, sendo ainda possível consultar o conjunto de URLs que levam a um dos URLs da tabela (próxima funcionalidade) através de um botão contido noutra coluna. É ainda possível avançar e recuar nas páginas da pesquisa caso existam mais do que 10 resultados.

Figura 6

Uma imagem com texto, captura de ecrã, computador, software

Descrição gerada automaticamenteExiste ainda uma página de erro que é mostrada sempre que um aconteça, seja este por parte do cliente ou por parte do servidor. Esta página mostra o código de erro do HTTP seguida de uma descrição do problema. A *Figura 7* mostra o erro que acontece ao tentar aceder à pesquisa que contém a palavra “ronaldo” na página “-1”.

Figura 7

## **Pesquisa de URLs que têm uma ligação para um dado URL**

Endpoint: <http://localhost:8080/search_url>

Uma imagem com texto, captura de ecrã, software, Página web

Descrição gerada automaticamenteUm utilizador pode também pesquisar que URLs tem uma hiperligação para um outro URL como descrito anteriormente. O utilizador pode aceder a este Endpoint onde se vai deparar com uma página semelhante à *Figura 5*. Os resultados são novamente apresentados em forma de tabela sendo possível mais uma vez avançar e recuar nas páginas de pesquisa (*Figura 8*).

Figura 8

Através de uma chamada RMI o WebServer pede ao Search Module os resultados da pesquisa. Este manda o resultado em forma de String que vai ser analisada e partida para poder ser colocada na tabela mostrada na figura anterior.

# Integração com a REST API do Hacker News

Duas das funcionalidades novas a serem desenvolvidas nesta fase do projeto envolviam a REST (Representational State Transfer) API do Hacker News. Uma REST API baseia-se nos princípios do protocolo HTTP. Uma API é constituída por um conjunto de Endpoints que podem ser acedidos através de uma solicitação GET, POST, PUT ou DELETE com os respetivos parâmetros necessários.

A nível de código é estabelecida uma conexão com a API e, através do método GET, extrair informação a ser acoplada no sistema. Caso exista algum erro com a conexão estabelecida ou o Endpoint não exista é devolvida uma página de erro semelhante a [*Figura 7*.](#f) As respostas vêm no formato de JSON e são analisadas e decompostas de forma a aproveitar a parte essencial da mesma.

Para o projeto foram feitas duas funcionalidades que extraem informação da API do Hacker News através de dois Endpoints.

## **Indexar as Top Stories do Hacker News que contenham um conjunto de termos**

Endpoint: <http://localhost:8080/top-stories>

É possível indexar as top stories que contenham um conjunto de termos no título da mensagem. As top stories são as páginas mais pesquisadas no Hacker News numa dada altura. Faz-se um GET Request à API com o URL [*https://hacker-news.firebaseio.com/v0/topstories.json?print=pretty*](https://hacker-news.firebaseio.com/v0/topstories.json?print=pretty) de forma a obter todas essas pesquisas mais comuns. A mensagem devolvida é uma lista de ID’s destas top stories. Obtém-se a informação de cada uma das stories e verifica-se se a palavra inserida na caixa de texto está contida nessa Top Storie. Essa informação é-nos dada pelo URL <https://hacker-news.firebaseio.com/v0/item/35958876.json?print=pretty> para o caso da Storie 35958876 por exemplo. Caso a palavra esteja contida no título de uma dada Storie, o URL correspondente a ela é enviado para o Search Module de forma a ser analisado pelos Downloaders e guardado nos Index Storage Barrels.

A *Figura 9* representa a página que mostra o resultado de indexar as Top Stories do Hacker News que contém a palavra “bird”.

Uma imagem com texto, captura de ecrã, software, Página web

Descrição gerada automaticamente

Figura 9

## **Indexar as Stories de um dado utilizador do Hacker News**

Endpoint: <http://localhost:8080/top-stories-user>

Fazendo um GET Request ao Endpoint <https://hacker-news.firebaseio.com/v0/user/john.json?print=pretty> recebe-se as Stories que, neste caso, o utilizador “john” publicou. Estas stories são acedidas à semelhança da funcionalidade anterior para serem indexadas. Uma vez que nem todas as stories contém um URL, podendo conter apenas uma mensagem, essas Stories são ignoradas em ambas as funcionalidades. A página mostrada na *Figura 10* mostra as Stories publicadas pelo utilizador “john” após serem adicionadas à Queue de URLs a serem indexados.

Uma imagem com texto, cartão de visita, Tipo de letra, captura de ecrã

Descrição gerada automaticamenteUma imagem com texto, captura de ecrã, software, computador

Descrição gerada automaticamenteNo caso do utilizador não existir é novamente mostrada uma mensagem de erro (*Figura 11*).

Figura 11

Figura 10

# Notificações em tempo real

Endpoint: <http://localhost:8080/stats>

Além da integração da API da Hacker News, foi proposto adicionar a funcionalidade de notificações em tempo real para os utilizadores. Para alcançar isso, foram utilizados WebSockets para estabelecer uma comunicação bidirecional em tempo real entre o cliente e o servidor.

O protocolo escolhido para suportar a comunicação WebSocket foi o SockJS, que é uma solução mais robusta em comparação ao WebSocket puro. O SockJS permite a adaptação em diferentes ambientes e melhora a compatibilidade entre os navegadores e servidores. Além disso, para facilitar a troca de mensagens, foi utilizado o protocolo STOMP (Simple Text Oriented Messaging Protocol), que define uma estrutura para a troca de mensagens entre clientes e servidores.

No código, é criada uma configuração no lado do servidor para configurar o broker de mensagens WebSocket. Esse broker é configurado para usar o destino “/stats” como canal de entrega das mensagens do servidor para os clientes. Além disso, é registado o Endpoint “/stats/update”, no qual os clientes podem se conectar para trocar mensagens em tempo real.

Um cliente WebSocket também é criado para enviar as mensagens. Quando esse cliente está ativo, ele conecta-se ao servidor e envia as estatísticas, que são redirecionadas para o Endpoint <http://localhost:8080/stats-update>. A partir desse Endpoint, as informações são enviadas para o frontend, onde o javascript é responsável por atualizar as informações no ecrã dos utilizadores que estão conectados a esse Endpoint (<http://localhost:8080/stats>).

Dessa forma, as notificações em tempo real são entregues aos clientes permitindo que as informações sejam atualizadas automaticamente na interface do utilizador, sem a necessidade de recarregar manualmente a página.

O exemplo apresentado (*Figura 12*) mostra como a página de estatísticas é apresentada, contendo a lista de Barrels ativos na esquerda, a lista de Downloaders ativos no centro e as palavras mais pesquisadas na direita. Como explicado anteriormente, todas estas estatísticas são atualizadas em tempo real sem que o utilizador necessite de recarregar a página manualmente, somente quando algum dado e alterado. No exemplo apresentado na *Figura 12* o Sistema estava a correr com um Barrel ativo e nenhum Downloader ativo.

Consecutivamente, na *Figura 13* mostra a página de estatísticas podendo visualizar que o conteúdo foi atualizado sem ter atualizado a página, podendo se visualizar no inspetor a estrutura de dados que foi enviada pelo WebSocket. Neste exemplo o sistema foi atualizado ao ligar um Downloader atualizando a interface de todos os utilizadores que estavam na página das estatísticas, ou seja, na página da (Figura 13).

Uma imagem com texto, captura de ecrã, software, design

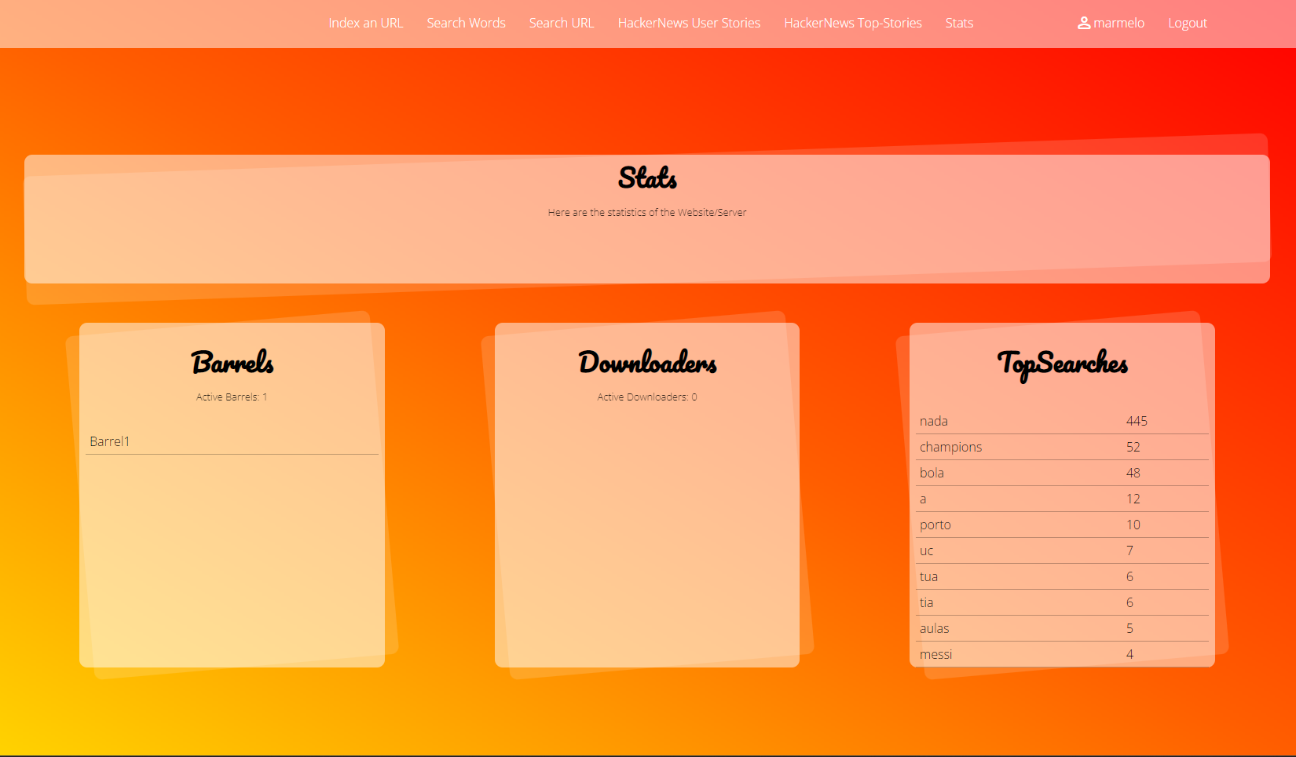
Descrição gerada automaticamente

Figura 13

Figura 12

# Alterações do Backend face à Meta 1

Para dar suporte ao WebServer desenvolvido nesta meta foram necessárias fazer ligeiras alterações aos componentes do backend.

A alteração feita ao backend foi a criação de uma nova interface (ControllerInterface.java) de forma a fazer a comunicação bidirecional entre o Search Module e o WebServer (callback). Esta interface é utilizada para fazer o *Susbcribe* e o *Unsubscribe* do WebServer no Search Module de forma a enviar a informação atualizada para o WebSocket.

# Testes Realizados

Teste: Correr o WebServer sem o Search Module ativo.

Estado: Passou

Descrição: Como era de esperar, nenhuma funcionalidade do site irá funcionar, pelo que é carregada uma mensagem de erro com o Status 500 (erro do lado do servidor).

Teste: Correr o WebServer com o Search Module e os restantes componentes ativos.

Estado: Passou

Descrição: Todas as funcionalidades funcionam como suposto, a não ser que aconteça algum erro por parte do utilizador.

Teste: Correr o WebServer com o Search Module ativo e sem nenhum Barrel.

Estado: Passou

Descrição: O cliente é capaz de fazer o login e indexar um link, seja ele manualmente ou por via da API do HackerNews. Estes URLs a serem analisados vão para a Queue de URLs, independentemente se existe algum Downloader ativo. Caso o utilizador tente fazer alguma pesquisa depara-se com um erro 503 dizendo que não há nenhum Barrel ativo.

Teste: Correr o WebServer com o Search Module ativo e sem nenhum Downloader.

Estado: Passou

Descrição: Como explicado anteriormente, as funcionalidades de indexação ficam em espera. Estes URLs ficam na Queue até que algum Downloader os analise.

Teste: Pesquisar um conjunto de termos que nenhum URL presente no Sistema os contenha.

Estado: Passou

Descrição: O utilizador depara-se com o erro 404, uma vez que a página de resposta à pesquisa por ele inserida não existe. A descrição mostrada na página de erro explica ao utilizador que não foram encontradas páginas com o conjunto de termos inserido por ele.

Teste: Forçar aceder a uma página que não existe através de um Endpoint fictício.

Estado: Passou

Descrição: Como visto anteriormente, o Endpoint das pesquisas tem como parâmetros as palavras da pesquisa e o número correspondente à página da pesquisa. Caso este número maior que o número de páginas existentes ou menor que 0, o cliente depara-se com uma página de erro 404.

Teste: Indexar as Stories de um utilizador do Hacker News que não existe.

Estado: Passou

Descrição: O Sistema informa o cliente que esse utilizador não existe através de uma página de erro.

# Conclusão

Ao longo do semestre aprendemos vários conceitos que nos vão acompanhar no resto do nosso percurso académico e, *quiçá*, em todo o nosso percurso profissional. Com este projeto pusemos esses conhecimentos em prática fazendo uso de uma framework (Spring), de uma API e de WebSockets.

Aproveitando todo o trabalho feito na primeira Meta, a integração com o Sprint Boot foi relativamente simples uma vez que grande parte do trabalho estava feito. O objetivo principal desta Meta de trabalho foi entregar uma Interface Gráfica através da Web ao utilizador para que este possa usar todas as funcionalidades desenvolvidas na Meta 1. O grande entrave desta fase do trabalho foi, sem dúvida, a implementação do WebSocket, uma vez que este conceito era algo de novo para nós. Em relação ao Spring Boot, a dificuldade tida foi no primeiro conjunto de Endpoints. A partir daí, construi-se um Endpoint para cada funcionalidade de forma semelhante a estes primeiros Endpoints, alterando apenas a análise da resposta recebida pelo Search Module.

Em suma, estamos contentes e orgulhosos do trabalho que desenvolvemos, pois, este tipo de trabalhos poe à prova aspetos como o espírito critico, o trabalho em equipa e a capacidade de recuperar de erros/bugs, aspetos estes que serão muito importantes no contexto profissional.