Googol: Motor de pesquisa de páginas Web

Meta 2

Sistemas distribuídos 2022/2023

David Marcelino Mendes Palaio - 2018283864 - uc2018283864@student.uc.pt

Liu Haolong - 2018288018 - uc2018288018@student.uc.pt

Índice

Introdução

Este projeto foi desenvolvido em função da primeira meta de maneira a que esta seja capaz de ter uma interface web com operações REST que permitam exercer as funcionalidades anteriormente implementadas. Também foi adicionado funcionalidades à base da API do Hacker News.

Arquitetura do software

Em relação à primeira meta, foram alterados alguns detalhes do projeto original: foram reduzidos o número de **Downloaders**, é apenas mantido um ativo e foram alterados alguns métodos RMI para serem adaptados ao cliente web.

O cliente web é implementado à base de Springboot e Thymeleaf. A ligação de RMI entre este e o search module é implementado no controlador do spring boot(classe **GreetingController**). É implementada uma classe **rmiHandler** para este fim com a notação **@Component** para possibilidade de dar iniciar automaticamente dentro do controlador com a notação **@Autowired**. A versão do java utilizada é 18.

```java
@Component
class rmiHandler{
    SearchModule_I h;
    Client c;

    public rmiHandler(){
        try {
            String ficheiro = "serverIp.txt";
            File myObj = new File(ficheiro);
            Scanner myReader = new Scanner(myObj);
            String data = myReader.nextLine();
            //System.out.println(data);
            myReader.close();
            String name = "rmi://" + data + ":8000/projeto";
            h = (SearchModule_I) Naming.lookup(name);
            c = new Client();
            h.subscribe(name:"WebClient", c);
        } catch (MalformedURLException | RemoteException | NotBoundException e) {
            e.printStackTrace();
        } catch (FileNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

    }
}
```

A estrutura do cliente web é simples, existem 7 endpoints que correspondem a 7 funcionalidades: registar utilizador, login do mesmo, indexar um certo URL, pesquisar, indexar top stories, indexar user stories e mostrar as estatísticas.

Register Login Add Url Search Search Top Stories Search User Stories Statistics

As credenciais do registo e do login do utilizador são guardadas localmente numa lista. Ao fazer o login, é atribuído um token que é necessário para as operações(apenas implementado no **Search**, onde obriga a inserir o token no pedido). Este token é criado com o **@SCOPE** SCOPE_SESSION, de forma a que seja renovada a cada sessão do utilizador. Pode entretanto ocorrer o caso de manter a sessão ativa e mudar de utilizador. Poderia resolver-se atribuindo um token(mantendo o mesmo scope) cada vez que se fazia um login.

Cada uma das ligações acima são pedidos HTML que são tratados pelo controlador, dado pela notação **@Controller**. A resolução para as várias ligações são semelhantes, um pedido **GET** é realizado e retorna para uma view, um ficheiro HTML, que contém um formulário que por sua vez passa o valor recebido para um **POST**, onde é armazenado e tratado conforme as funcionalidades. Toda a informação que é mostrada nas views é armazenada num objeto **Model**, que pode ser alterado ao longo do funcionamento do programa. A view mais complexa foi a apresentação de resultados de pesquisa, que inclui dois toggles: um para mostrar os Urls que apontam para o Url da pesquisa e outro para mostrar mais resultados.

```
function toggleResults() {
    results.forEach((result, index) => {
        if (index < visibleResults) {
            result.style.display = 'block';
        } else {
            result.style.display = 'none';
        }
    });
}

function toggleUrls() {
    toggleBtns.forEach(btn => {
        btn.addEventListener('click', () => {
            const urlsContainer = btn.nextElementSibling;
            if (urlsContainer.style.display === 'none') {
                urlsContainer.style.display = 'block';
            } else {
                urlsContainer.style.display = 'none';
            }
        });
    });
}

function loadMore() {
    visibleResults += 10;
    if (visibleResults >= totalResults) {
        loadMoreBtn.style.display = 'none';
    }
    toggleResults();
}

loadMoreBtn.addEventListener('click', loadMore);

toggleResults();
toggleUrls();
```

Em relação à ligação com a API de Hacker News, foi utilizado o objeto **RestTemplate** para consumir as operações REST da API anterior. Utilizaram-se dois **record** para armazenar a informação: um para guardar os itens e outro para os utilizadores. É necessário notar que na pesquisa de Top Stories os itens que tinham texto que correspondiam à query do utilizador mas que não incluíam Urls não são indexados.

As websockets não foram implementadas. Apesar da funcionalidade de mostrar as top 10 pesquisas estar ativa, esta não atualiza em tempo real a página, apenas enviando outro pedido HTML.

Funcionamento do Software

Para o funcionamento do programa em todo, é necessário que seja primeiramente executado o servidor RMI, **SearchModule**, depois o **Barrel** e posteriormente o **Downloader**. Para aceder às funcionalidades basta inicializar o cliente ou cliente web(abrir browser e ir para o ip indicado no ficheiro serverIp na porta 8080).

As funcionalidades de pesquisa de Hacker News podem levar algum tempo, dado à quantidade de Urls que já não são acessíveis e à quantidade de itens que tem que pesquisar no caso dos Top Stories. Nesta última poderia-se ter implementado um cache para melhorar o tempo de pesquisa.

.

Testes feitos à plataforma

Apenas foram feitos testes às funcionalidades do programa, introduzindo valores corretos e incorretos e os resultados foram dentro dos previstos.