FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

Sistemas Distribuídos

Googol - Meta 2

Dinu Nicolae Bosîi – 2019237103

2022/2023

Índice

1.Introdução

A segunda meta deste projeto tem como objetivo a implementação de uma interface Web que comunique com o Servidor RMI implementado na primeira meta. Para além disso, é necessário implementar funcionalidades para comunicação assíncrona entre clientes e o servidor bem como a inter«gração com outros serviços REST.

2.Arquitetura de Software

A aplicação desenvolvida nesta meta consiste numa aplicação Spring Boot com integração do código da meta anterior. Na pasta "src" contém duas packages, para diferenciar o código da meta1 com o da nova meta2.

Existem 9 ficheiros no package 'com.example.demo' dos quais os mais importantes são:

- **DemoApplication.java:**

  Contém a função main que corre a aplicação desenvolvida, como se pode observar pela anotação *@SpringBootApplication.* Contém também a anotação *@EnableScheduling* para possibilitar a execução periódica do método que se encontra no ficheiro adminPageInfo.

- **AdminObject.java:**
  Classe que define os objetos para a apresentação do conteúdo da página de administração.

- **adminPageInfo.java:**

  Responsável pela atualização da página de administração: num objeto da classe anterior, é armazenado a informação a expor na página de administração, sendo esta informação obtida por RMI. A informação é obtida periodicamente através com auxílio à componente *@Service* fornecida pelo *SpringBoot*. No caso em que a informação é atualizada, é enviado o objeto para a página de administração através de um *WebSocket*.

- **SearchModule.java:**

  Código do servidor da meta1, onde se atualiza um objeto da classe *AdminObject* para ser acedido pela componente da classe de *adminPageInfo*.

- **ClientController.java:**
  Controlador principal da aplicação, responsável pelas funcionalidades principais do motor de busca à exceção da página de administração. Contém um *endpoint* para cada funcionalidade, responsáveis por responder a pedidos http, redirecionando para a página html adequada (com recurso a Thymeleaf) e com a informação processada.

- **AdminController.java:**
  Responsável pelo *endpoint* da página de administração. Obtém por RMI a informação antes de servir a página html, dado que, se não houver alterações ao conteúdo da página, não é obtida a informação.
- **WebSocketConfig.java:**
  Defina a configuração dos Websockets para a página de administração. Contém um Message broker para lidar com o envio de mensagens e um endpoint Stomp que define o método para a troca de mensagens entre os clientes e o servidor. O endpoint foi definido para /my-websocket com opção de fallback através de um SockJS que tem um comportamento semelhante ao WebSocket.

Para além disso, encontram-se na pasta *resources* as tanto as páginas html e css usados (desenvolvidos com recurso a ChatGPT) e o ficheiro app.js para a conexão de clientes ao WebSocket e subscrição ao tópico por onde irão receber informação da página de administração.

3. Integração do Spring Boot com o Servidor RMI

Como foi mencionado anteriormente, a aplicação desenvolvida consiste numa aplicação de Spring Boot com o código do Servidor RMI no seu interior, por motivos de conveniência, sendo independentes na forma como se corre ambos os programas.

4.Integração com serviço REST

Com recurso ao código disponibilizado para a implementação das funcionalidades relativas ao Hacker News, são implementados dois endpoints principais, que podem ser encontrados no ClientController.

Para a indexação de URLs das top stories que contenham os termos de uma pesquisa, é usado um botão após uma pesquisa genérica (a partir da página inicial) que redireciona para o endpoint */indexTopStories* com a query pesquisada. São obtidas as 500 Top Stories mais recentes, e comparando com as Top 100 por motivos de performance, verifica-se quais destas contêm os termos de pesquisa sendo posteriormente indexadas através do Servidor RMI.

O endpoint "/hackerNewsUserSearch", segue a mesma lógica que o anterior, sendo no entanto acedido pelo endpoint "/hackerNewsUser"(acedido pelo menu no top de qualquer página da aplicação). Escolhido um utilizador, obtém-se a sua informação a partir de um JSON que fica armazenado dentro de um objeto do tipo "HackerNewsUserRecord", contendo uma componente "submitted" com os ids de todos os seus comentários/stories/polls. Através do id da story, obtém-se um objeto do tipo "HackerNewsUserRecord", que contêm as suas informações possibilitando a indexação do seu Url caso este exista.

5. WebSockets

Para possibilitar a implementação da página de administração e de modo que o conteúdo desta seja atualizado em tempo real, são configurados WebSockets. Para cada cliente é aberto um canal de comunicação através de um WebSocket e com recurso ao componente dentro de adminPageInfo, é enviada a informação necessária pelo canal.

Após a obtenção da informação (por RMI), esta é enviada com recurso a um "SimpMessagingTemplate", que permite o envio de objetos. Através de um message broker, as mensagens são enviadas para todos os clientes conectados ao servidor. Do lado do cliente, corre o JavaScript contido no ficheiro "resources/static/app.js", que permite conectar ao WebSocket (no endpoint "/my-websocket") e subscrever ao tópico "/admin/info", por onde receberá mensagens no formato JSON, sendo a sua informação apresentada na página de html "administration.html". Sempre que há alteração na informação, a função handleUpdate() é responsável por alterar a informação apresentada no cliente.

6.Testes

Indexação de um URL	**Pass**
Pesquisa de um conjunto de termos	**Pass**
Resultados agrupados de 10 em 10	**Pass**
Resultados ordenados	**Pass**
Lista de ligações para um URL	**Pass**
Página de administração	**Pass**
Página de administração atualizada em tempo real	**Pass**
Indexação das stories de um user	**Pass**
Indexação das Top Stories com os termos da pesquisa	**pass**

7.Conclusão

O desenvolvimento deste projeto permitiu adquirir conhecimento aprofundado sobre o funcionamento de sistemas distribuídos, bem como o desenvolvimento de uma interface Web através da framework Spring Boot e a sua integração com um Servidor Web.

As funcionalidades principais desta meta foram todas implementadas, apesar de várias melhorias que poderiam melhorar significativamente a performance do motor de busca (não foi implementado caching das pesquisas efetuadas, o que terá um impacto na performance para bases de dados de maior tamanho). A página de administração, apesar de funcional, poderia ter sido implementada de forma a que as informações fossem apresentadas ao cliente de forma mais rápida (com o servidor RMI a enviar a informação sempre que é detetada uma alteração, em vez de ter outro serviço a fazer uma verificação simultânea).