FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D COIMBRA

Googol: Motor de pesquisa de páginas Web

Sistemas Distribuídos 2022/2023

Meta 2

Licenciatura em Engenharia Informática

Trabalho realizado por:
Bruno Eduardo Machado Sequeira nº 2020235721
Rui Pedro Capelas Santos nº 2020225542

Introdução:

Nesta segunda meta do projeto, iremos criar um FrontEnd Web para a aplicação Googol que foi implementada na meta 1.

Com isso, os utilizadores podem aceder ao serviço a partir de quase qualquer dispositivo com internet, sem necessitar de instalar nenhum software de cliente.

Para que os utilizadores web possam aceder à mesma informação que os utilizadores na aplicação desktop usaremos um servidor RMI que foi desenvolvido na meta 1.

Portanto os utilizadores deverão ter as mesmas funcionalidades que foram desenvolvidas na meta 1, como por exemplo, indexar URLs, pesquisar por termos, ver as estatísticas, etc.

Visto que o aspecto interativo é muito importante na Web, a aplicação deverá mostrar algumas alterações em tempo real, como por exemplo, Barrels e Downloaders ativos, e o top10 de pesquisas realizadas pelos utilizadores Web.

Para que a aplicação tenha uma funcionalidade mais rica iremos integrar a nossa aplicação com a API do Hacker News, através de APIs Rest. Após realização de uma pesquisa, será possível ir buscar URL's às Top Stories do Hacker News que contenham os termos pesquisados no Googol.

Deverá também ser possível a um utilizador com conta no HackerNews pedir ao Googol para ir buscar todas as suas "stories" e indexar todos os seus URLs do Hacker News.

Iremos apresentar a Arquitetura de Software implementados, a implementação do Spring Boot, a implementação dos Websockets, do serviço REST e os testes realizados ao longo da meta2.

Arquitetura de Software:

Nesta nova versão da aplicação, para além dos programas da meta 1 (Downloaders, Index Storage Barrels, Search Module, Clientes, Queue_url) temos também os programas da nossa aplicação Spring boot (googol, Message, MessagingController e WebSocketConfig) e os programas para fazer a comunicação com a API hackernews (HackerNewsItemRecord, HackerNewsUserRecord e MyHackerNewsController).

Foi seguida uma abordagem MVC (model-view-controller). Os models representam os dados e a sua manipulação, as views representam a informação, neste caso numa página web e o controller serve de intermediário entre os models e as views, recebendo pedidos, processando-os e determinando a resposta mais apropriada. Ou seja, escolhe a view mais apropriada e passa-lhe os dados necessários.

Na nossa aplicação temos dois controladores, um para tratar da comunicação com a hackerNews API (MyHackerNewsController) e outro para tratar do web server (MessagingController). Nesses controladores algumas das funções têm como parâmetro o model que vai ser o que vamos então usar para passar alguns parâmetros às nossas views estão guardadas na pasta resources e correspondem a páginas web (uma por cada ponto a desenvolver no trabalho, status, login, register, search, linkedpages, indexing, hacker News Indexing). Temos também 3 classes que usamos para obter informação que nos é fornecida pelo utilizador e estão na pasta forms.

Também fazemos uso de websockets para fazer comunicação entre o webserver e uma página da nossa aplicação (detalhes mais à frente). Iremos falar dos seguintes pontos:

- Serviço REST.
- Websockets
- Spring Boot

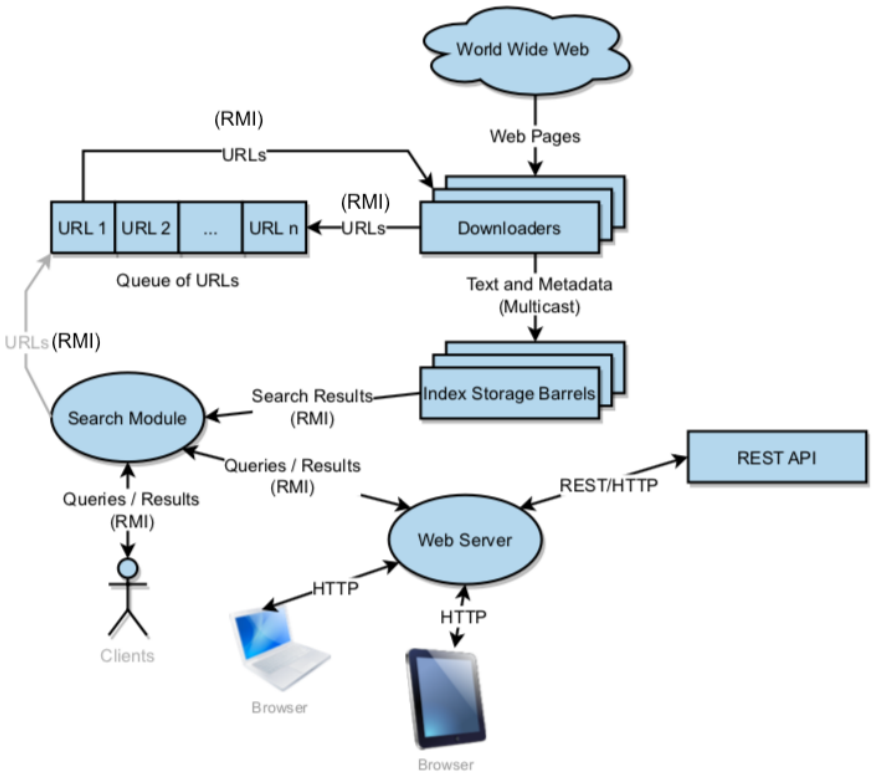

Figura - Arquitetura de Software.

Spring Boot:

Para podermos utilizar o spring boot juntamente com a nossa aplicação desenvolvida na meta1 foram necessários alguns ajustes nomeadamente no search module.

No cliente desenvolvido na meta 1, quando ele queria saber as estatísticas do servidor, era ativada uma thread que sempre que existisse alguma atualização no número de barrels e/ou downloaders as estatísticas seriam atualizadas em tempo real.

Na página de administração do webserver é o próprio webserver que está a pedir ao search module a ver se há atualização ou não. Isto foi resolvido facilmente com uma variável booleana no lado do searchmodule que é posta a true se houver alterações e o webserver consulta essa variável e se for true pede a lista de downloaders e de barrels e mete a variável a false, se for false nao faz nada. O webserver comunica com o searchmodule via RMI por isso também tivemos de dar update ao MessageServerInterface, adicionando lá as funções getBarrels(), getDownloaders, getUpdated() (updated é a variável que indica se há atualização ou não) e setUpdated().

No que toca aos outros pontos como a indexação e as pesquisas foi só aproveitar as funções que já tínhamos da meta 1.

WebSockets:

Os websockets são uma tecnologia que permite a comunicação bidirecional em tempo real entre um servidor web e um cliente. Portanto, como pedido, implementamos os websockets para atualizar uma página de status em tempo real.

No método “Start” do controller, o servidor recebe uma solicitação para a ligação ‘/stats’. Ele obtém a sessão atual do cliente, sendo que o nome do usuário é recuperado e adicionado ao modelo, juntamente com o estado de logado.

Em seguida temos o método “onMessage” que é anotado com “@MessageMapping(“/message”)”, que significa que ele será chamado quando uma mensagem for recebida na ligação “/app/message” pelo servidor. O método recebe um Objeto “Message” e retorna uma “CompletableFuture”que fornece uma resposta ao cliente em formato assíncrono.

Dentro do método “onMessage”, há uma lógica para processar a mensagem recebida. Se for a primeira, logo aparece como “first”(Conexão do Cliente), o servidor obtém informações dos Downloaders, barrels e do top10 através do RMI. Sendo que estas são formatadas em uma String e retornadas como uma Message para o cliente.

Se a mensagem recebida não for “first” e a informação no search module estiver atualizada, fazemos o mesmo processo, ou seja, obtemos as informações e formatamos novamente, enviando como Message para o cliente.

No lado do cliente, temos funções JavaScript que lidam com a conexão aos websockets. A função “connect” é chamada quando a página é carregada e cria uma conexão com o servidor usando a Biblioteca Stomp.js.

Ao se conectar com sucesso, o cliente envia uma mensagem “first” para o servidor usando a função “sendMessageFirst”. Em seguida, ele se inscreve no tópico “/topic/messages” para receber mensagens do servidor.

Quando uma mensagem é recebida, a função “showMessage” é chamada. Ela formata e exibe as informações recebidas nas tabelas na página, antes de enviar para as tabelas, apagamos toda a informação que está na página.

A mensagem é formatada desta maneira:

“(y) - IP: x.x.x.x Porta:xx,(y+1) - IP: x.x.x.x Porta:xx,(y) - IP: x.x.x.x Porta:xx,(y+1) - IP: x.x.x.x Porta:xx||(y) - IP: x.x.x.x Porta:xx,(y+1) - IP: x.x.x.x Porta:xx,(y) - IP: x.x.x.x Porta:xx,(y+1) - IP: x.x.x.x Porta:xx||1-universidade - 10::2-escola-8”

Basicamente, temos e string entre os “||”, a primeira corresponde ao Downloader, a segunda aos Barrels e a 3ª são os top10 de pesquisas. Downloaders e barrels dividem-se em “,” e os top10 dividem-se em “::”.

Além disso, temos o “Connect”, “Disconnect” e “Send” que são associadas às respectivas funções para controlar a conexão com o servidor e enviar mensagens. Esta implementação de websockets permite que a página de status seja atualizada em tempo real conforme novas informações são recebidas do servidor, fornecendo assim uma experiência interativa com o utilizador.

Serviço REST:

De forma a integrar a aplicação com o serviço REST para a comunicar com a API do hacker news criamos o controller “MyHackerNewsController”. Nele temos 3 funções para lidar com as duas funcionalidades pedidas.

Para a funcionalidade “indexar URLs das top stories que contenham os termos da pesquisa” temos a função hackerNewsTopStories() que basicamente pede a API a lista de stories e verifica quais stories dessa lista contém os tokens de uma pesquisa. Depois para essas stories indexamos o url onde se encontram. De notar que apenas consideramos stories com texto e com um url associado. Como isto é um processo lento (+500 stories) consideramos apenas no máximo 10 stories que tenham os tokens de uma pesquisa.

Testes:

Teste 1	Login - Search Module disponível
Objetivo	Um utilizador deve conseguir dar login quando o Search Module está disponível.
Resultado obtido	Quando o cliente entra na página de login “/login” é-lhe solicitado para inserir o username e a password. Após inserir o que é pedido o login é feito com sucesso, sendo redirecionado para a página inicial.
Resultado	Passou

Teste 2	Login - Search Module indisponível
Objetivo	Um utilizador não deve conseguir dar login quando o Search Module está indisponível.
Resultado obtido	Quando o cliente entra na página de login “/login” é-lhe solicitado para inserir o username e a password. Após inserir o que é pedido, o login não é efetuado, aparecendo assim um Pop-up a dizer que o Search Module está indisponível.
Resultado	Passou

Teste 3	Registo - Search Module disponível
Objetivo	Deve ser possível registrar um novo utilizador quando o Search Module está disponível.
Resultado obtido	Quando o cliente entra na página de registo "/registry" ' é-lhe solicitado para inserir o username e a password. Após inserir o que é pedido o registo é feito com sucesso, sendo redirecionado para a página inicial.
Resultado	Passou

Teste 4	Registo - Search Module indisponível
Objetivo	Não deve ser possível registrar um novo utilizador quando o Search Module está indisponível.
Resultado obtido	Quando o cliente entra na página de registo "/registry" ' é-lhe solicitado para inserir o username e a password. Após inserir o que é pedido, o registo não é efetuado, aparecendo um Pop-up a dizer que o Search está indisponível.
Resultado	Passou

Teste 5	Indexação de um novo Url - Search Module Disponível
Objetivo	Deve ser possível um cliente adicionar um novo url para indexar
Resultado obtido	O cliente seleciona a opção para indexar um novo url página “/indexing”, indica esse url e ele é adicionado a queue com sucesso, e é redirecionado para a página inicial.
Resultado	Passou

Teste 6	Indexação de um novo Url - Search Module Indisponível
Objetivo	Não deve ser possível um cliente adicionar um novo url para indexar.
Resultado obtido	O cliente seleciona a opção para indexar um novo url página “/indexing”, indica esse url, não sendo adicionado à queue, aparecendo um Pop-up a dizer que o Search Module está indisponível, e é redirecionado para a mesma página “/indexing”
Resultado	Passou

Teste 7	Pesquisa de um token - Search Module Indisponível
Objetivo	Um cliente não consegue pesquisar todos os sites onde aparece um certo token.
Resultado obtido	O cliente seleciona a opção de pesquisar tokens (página inicial), coloca os tokens e aparece a mensagem “Search Module Indisponível"
Resultado	Passou

Teste 8	Pesquisa de um token 1 - Search Module Disponível
Objetivo	Um cliente pode pesquisar todos os sites onde aparece um certo token.
Resultado obtido	O cliente seleciona a opção de pesquisar tokens (página inicial) e aparecem todos os sites onde aparece aquele token
Resultado	Passou

Teste 9	Pesquisa de um token 2 - Search Module Disponível
Objetivo	Se um cliente inserir um token que ainda não foi encontrado, aparece uma mensagem a dizer que não há informação sobre esse token.
Resultado obtido	O cliente seleciona a opção de pesquisar tokens (página inicial) e aparece a mensagem “Sem resultados encontrados!!”
Resultado	Passou

Teste 10	Pesquisa de um token 3 - Search Module Disponível
Objetivo	Se um cliente inserir um token para pesquisa e não houver barrels disponíveis recebe uma mensagem a informar disso
Resultado obtido	O cliente seleciona a opção de pesquisar tokens (página inicial) e aparece a mensagem”Barrels Indisponíveis!!”
Resultado	Passou

Teste 11	Lista de páginas com ligação para uma pagina 1 - Search Module Indisponível
Objetivo	Um cliente não consegue ver a listagem das páginas com ligação a uma página que ele indica.
Resultado obtido	O cliente seleciona a opção de listagem de páginas (página “/linked”), coloca o link e aparece um Pop-up a dizer que o Search Module está indisponível.
Resultado	Passou

Teste 12	Lista de páginas com ligação para uma pagina 1 - Search Module Disponível
Objetivo	Um cliente pode pedir para listar as páginas com ligação a uma página que ele indicar
Resultado obtido	O cliente seleciona a opção de listagem de páginas (página “/linked”), coloca o link e aparece a lista de links que apontam para o link indicado, se não houver nenhum ou não existir nenhum barrel ativo aparece a mensagem ”Sem resultados”.
Resultado	Passou

Teste 13	Lista de páginas com ligação para uma pagina 2 - Search Module Disponível
Objetivo	Para qualquer link resultante da pesquisa de um token, deve ser possível a um cliente logado listar as páginas que apontam para ele.
Resultado obtido	O cliente faz a pesquisa de um token (página inicial) e aparece a listagem dos links. Depois o cliente pode colocar clicar em “See” por baixo do link que quer que seja mostrada a listagem e esta aparece.E pode clicar em “Don´t See” para que esta não seja mostrada.
Resultado	Passou

Teste 14	Lista de páginas com ligação para uma pagina 3 - Search Module Disponível
Objetivo	Para qualquer link resultante da pesquisa de um token, deve ser possível a um cliente não logado não mostrar a opção de ver a listagem.
Resultado obtido	O cliente faz a pesquisa de um token (página inicial) e aparece a listagem dos links, não aparecendo a opção “See” por baixo dos links.
Resultado	Passou

Teste 15	Consulta das estatísticas do servidor
Objetivo	Um cliente pode pedir para consultar as estatísticas do servidor
Resultado obtido	O cliente seleciona a opção Estatísticas “Admin” a página correspondente é “/stats” e aparecem o número de downloaders e barrels com os seus ip’s e portas. Além disso, aparece também o top 10 de tokens mais pesquisados.
Resultado	Passou

Teste 16	Consulta das estatísticas do servidor
Objetivo	Um cliente pode pedir para consultar as estatísticas do servidor
Resultado obtido	O cliente seleciona a opção Estatísticas “Admin” a página correspondente é “/stats” e aparecem o número de downloaders e barrels com os seus ip’s e portas. Além disso, aparece também o top 10 de tokens mais pesquisados.
Resultado	Passou

Teste 17	Indexar Urls das top stories que contenham os termos da pesquisa - Search Module Indisponível
Objetivo	Um cliente não consegue indexar URLS que contenham os termos da pesquisa.
Resultado obtido	O cliente vai à página inicial (“/search”), coloca os termos para a pesquisa e o resultado que aparece é que “Search Module Indisponível”.
Resultado	Passou

Teste 18	Indexar Urls das top stories que contenham os termos da pesquisa - Search Module Disponível
Objetivo	Um cliente consegue indexar URLS que contenham os termos da pesquisa.
Resultado obtido	O cliente vai à página inicial (“/search”), coloca os termos para a pesquisa e o resultado que aparece são os links encontrados e aparece uma “box” azul que se clicar indexa os urls das top stories que contenham os termos da pesquisa. (resultados aparecem no terminal)
Resultado	Passou

Teste 19	Indexar todas as stories de um utilizador - Search Module Indisponível
Objetivo	Um cliente não consegue indexar as stories de um utilizador.
Resultado obtido	O cliente vai à página (“/hackernewsindex”), colocando o username o resultado que aparece é um Pop-up a dizer que o Search Module está indisponível.
Resultado	Passou

Teste 20	Indexar todas as stories de um utilizador - Search Module Disponível
Objetivo	Um cliente consegue indexar URLS que contenham os termos da pesquisa.
Resultado obtido	O cliente vai à página inicial (“/hackernewsindex”), colocando o username o resultado sendo que este irá processar a informação colocando assim na Queue todos os urls.
Resultado	Passou

Testes relativamente ao servidor, realizados tanto na meta1 como na meta2.

Teste 21	Mandar um barrel abaixo e reiniciá-lo
Objetivo	Quanto um barrel vai abaixo e é, mais tarde, reiniciado, deve ser atualizado com a informação perdeu enquanto foi abaixo.
Resultado obtido	O barrel recebe com sucesso a informação perdida enquanto foi abaixo
Resultado	Passou

Teste 22	O programa vai abaixo - queue
Objetivo	Nesta situação, a queue deve conseguir ir buscar os urls que faltavam processar, quando o programa voltar ao ativo.
Resultado obtido	Quando o programa volta, a queue consegue com sucesso ir à base de dados buscar os urls por processar.
Resultado	Passou

Teste 23	Barrel entra a meio da mensagem multicast
Objetivo	Um barrel, mesmo que entre a meio da mensagem multicast dos downloaders para os barrels, deve receber tudo o que ficou para trás
Resultado obtido	Quando o barrel entra a meio da mensagem multicast, recebe tudo o que ficou para trás.
Resultado	Passou

Teste 24	Barrels vão abaixo -> Comportamento dos downloaders
Objetivo	Se durante o funcionamento do programa, os barrels forem todos abaixo, os downloaders devem parar e aguardar pela chegada de um barrel
Resultado obtido	Quando os barrels vão abaixo os downloaders aguardam a chegada de barrels para continuarem a trabalhar.
Resultado	Passou

Teste 25	Entrada de um barrel novo(sem info nenhuma associada a ele)
Objetivo	Se entrar um barrel novo, ele deve ir buscar à bd toda a informação que os outros barrels tinham
Resultado obtido	Quando o barrel novo entra, vai à bd buscar a informação que precisa.
Resultado	Passou

Teste 26	Restart de um barrel
Objetivo	Se um barrel for abaixo e reiniciar e se, ou é o unico ativo, ou os restantes barrels ativos não têm toda a informação que precisa nas hashmaps deles, vai à bd buscar o que precisa.
Resultado obtido	Quando o barrel entra, vai à bd buscar a informação que precisa.
Resultado	Passou

Teste 27	Barrel vai abaixo a meio de pedido de cliente
Objetivo	Se um barrel vai abaixo enquanto está a responder a um pedido de um cliente, esse pedido deve ser transferido para outro barrel ativo.
Resultado obtido	Quando um barrel vai abaixo, o pedido do cliente é transferido para outro cliente.
Resultado	Passou

Teste 28	Verificar a entrada de um barrel
Objetivo	Quando queremos adicionar um barrel, temos de lhe fornecer um id, por isso, temos de garantir que não há dois barrels ativos com o mesmo id.
Resultado obtido	Quando é adicionado um barrel, se tiver um id igual a um dos barrels já existentes ativos é lhe negada a entrada.
Resultado	Passou

Teste 29	Verificar a entrada de um downloader
Objetivo	Quando queremos adicionar um downloader, temos de lhe fornecer um id, por isso, temos de garantir que não há dois downloaders ativos com o mesmo id.
Resultado obtido	Quando é adicionado um downloader, se tiver um id igual a um dos downloader já existentes ativos é lhe negada a entrada.
Resultado	Passou

Conclusão:

Após realização da meta 2, ficamos felizes com o resultado obtido, visto que os objetivos foram completados, e que apesar de tudo achamos a ideia deste projeto muito engraçada e importante para o nosso conhecimento.

Tivemos dificuldade nas novas tecnologias, por exemplo, o uso de SpringBoot nunca usamos essa tecnologia no nosso caminho enquanto estudantes, tal como os websockets, o que nos desafiou a aprender para implementação deste projeto.

Em relação à meta1 esta meta demorou menos tempo, visto que a ideia toda já estava implementada, apenas bastava ajustar para o SpringBoot, implementar os WebSockets, etc.

Concluindo, no nosso ponto de vista, este projeto (meta1 e meta2) permitiu-nos que as nossas capacidades, em criar uma camada de acesso aos dados usando Java RMI, usar sockets Multicast, programar um sistema pesquisa de páginas Web com uma arquitetura cliente-servidor, seguir um modelo multi thread para desenhar os servidores, em desenvolver uma interface Web para aplicação, ter integrado a interface com a aplicação desenvolvida na primeira meta, aplicado tecnologias de programação Web principalmente SpringBoot, em aplicar o s Websockets para comunicar assincronamente com os clientes, ter integrado a aplicação com os serviços REST externos e garantir a disponibilidade da aplicação aumentaram.

Biografia:

https://www.facebook.com/DEI.UC/
https://gist.github.com/sebleier/554280
https://gist.github.com/alopes/5358189
https://www.uc.pt/identidadevisual/Marcas_UC_submarcas
Materiais da Cadeira.