Googol: Motor de pesquisa de páginas Web

Sistemas Distribuídos 2022/23 — Meta 2 — 17 de maio de 2023 (21:59)

Resumo

Este projeto tem como objetivo criar um motor de pesquisa de páginas Web. Pretende-se que tenha um subconjunto de funcionalidades semelhantes aos serviços Google.com, Bing.com e DuckDuckGo.com, incluindo indexação automática (*Web crawler*) e busca (*search engine*). O sistema deverá ter todas as informações relevantes sobre as páginas, tais como o URL, o título, uma citação de texto e outras que considere importantes. Ao realizar uma busca, um utilizador obtém a lista de páginas que contenham as palavras pesquisadas, com as demais informações. Os utilizadores podem sugerir URLs para serem indexados pelo sistema. Partindo desses URLs, o sistema deve indexar recursivamente todos as ligações encontradas em cada página.

1 Objetivos do projeto

No final do projeto cada estudante deverá ter:

- Desenvolvido uma interface Web para a aplicação Googol.
- Ter integrado a interface Web com a aplicação desenvolvida na primeira meta.
- Aplicado tecnologias de programação para a Web, designadamente Spring Boot.
- Seguido a arquitetura MVC para desenvolvimento Web.
- Aplicado WebSockets para comunicar assincronamente com os clientes.
- Integrado a aplicação com serviços REST externos.

2 Visão geral

Nesta segunda meta do projeto, os estudantes irão criar um *frontend* Web para a aplicação Googol. Esta nova interface irá possibilitar que os utilizadores acedam ao serviço a partir de quase qualquer dispositivo com Internet no planeta, sem necessitar de instalação de software cliente. Como a interoperabilidade é um requisito muito importante, utilizadores Web deverão aceder à mesma informação que os utilizadores na aplicação desktop. Para tal, o servidor Web deverá usar o servidor RMI desenvolvido na meta 1.

Os utilizadores deverão ter as mesmas funcionalidades através da aplicação Web que estavam disponíveis na meta 1. Portanto, a interface Web deverá indexar URLs,

pesquisar páginas, etc. Como o aspeto interativo é muito importante na Web, a aplicação deverá mostrar algumas alterações em tempo real, nomeadamente as informações gerais sobre o sistema (downloaders e barrels ativos, etc.). Como os utilizadores estão cada vez mais exigentes, técnicas menos robustas como meta-refresh e iframes ocultas não serão consideradas.

Finalmente, as aplicações de hoje em dia são integradas entre si para se conseguir funcionalidade mais rica. Através de APIs REST, irão integrar a vossa aplicação com a API do Hacker News. Na sequência de uma pesquisa, deverá ser possível ir buscar URLs às "top stories" do Hacker News que contenham os termos pesquisados no Googol. Deverá também ser possível a um utilizador com conta no Hacker News pedir ao Googol para ir buscar todas as suas "stories" e indexar todos os seus URLs do Hacker News.

3 Arquitetura

A Figura 1 mostra a arquitetura geral do projeto. Os elementos adicionados em relação à meta 1 referem-se à segunda meta do projeto. O servidor Web deverá ligar-se por RMI ao servidor de dados, garantindo a interoperabilidade com os clientes da primeira meta.

Devem desenvolver uma aplicação Web que corra num servidor HTTP e que atue como um cliente do servidor RMI. Os utilizadores irão usar Web browsers para se ligarem ao servidor Web e pedirem páginas através do protocolo HTTP. Para melhorar a experiência de utilização poderão ponderar fazer alguns dos pedidos via AJAX em vez de carregar a página toda. A comunicação em tempo real para o browser deverá ser construída usando WebSockets.

4 Interface Web

Pretende-se criar uma aplicação Web que disponibilize as mesmas funcionalidades da meta 1 através da Internet. Usando uma arquitetura MVC, deverão implementar os seguintes requisitos funcionais usando Spring Boot com Thymeleaf:

1. **Indexar novo URL.** Um utilizador deve poder introduzir manualmente um URL para ser indexado. Esse URL será então visitado por um Downloader (*Web crawler*) e ficará associado, no índice invertido, às palavras que forem encontradas no texto.

2. **Indexar recursivamente todos os URLs encontrados.** O indexador automático deve visitar os URLs que forem encontrados em páginas previamente visitadas. Assim, o índice é construído recursivamente. Sugere-se a utilização de uma fila de URLs para este efeito.

3. **Pesquisar páginas que contenham um conjunto de termos.** Qualquer utilizador deve poder realizar uma pesquisa. Para tal, o motor de busca consulta o índice invertido e apresenta a lista de páginas que contêm todos os termos da pesquisa. Para cada resultado da pesquisa, deve mostrar o título da página, o URL completo e uma citação curta composta por texto da página. Os resultados da pesquisa devem ser paginados de 10 em 10, tal como os motores de busca habituais.

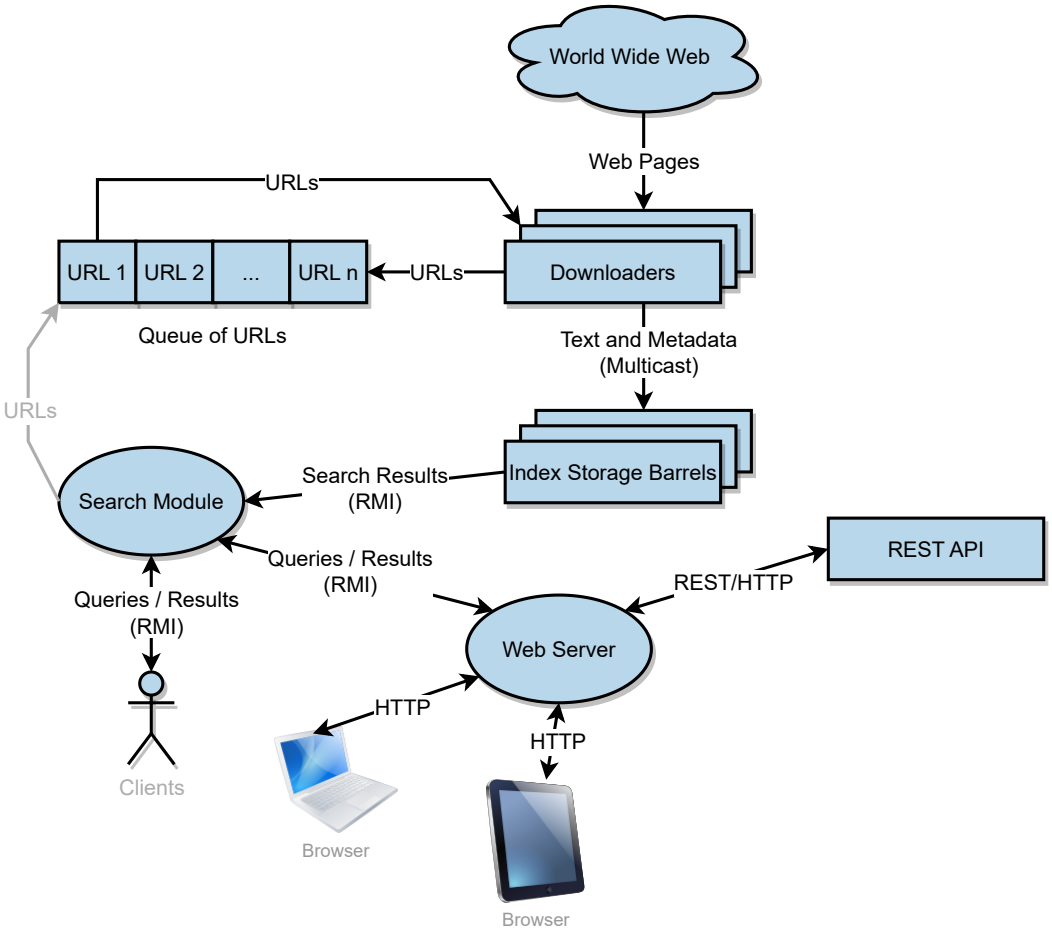

Fig. 1: Arquitetura da aplicação, incluindo os componentes da Meta 1.

4. **Resultados de pesquisa ordenados por importância.** Os resultados de uma pesquisa (funcionalidade anterior) devem ser apresentados por ordem de relevância. Para simplificar, considera-se que uma página é mais relevante se tiver mais ligações *de* outras páginas. Assim, o indexador automático deve manter, para cada URL, a lista de outros URLs que fazem ligação para ele.

5. **Consultar lista de páginas com ligação para uma página específica.** É possível saber, para cada página resultante de uma pesquisa, todas as ligações conhecidas que apontem para essa página. Esta funcionalidade pode estar associada à funcionalidade de pesquisa (por exemplo, uma opção associada a cada resultado).

6. **Página de administração atualizada em tempo real.** Todos os utilizadores têm acesso a uma opção de consulta de informações gerais sobre o sistema. Esta informação será atualizada apenas quando houver alterações. Pretende-se saber o estado do sistema, designadamente a lista de Downloaders e de Barrels ativos (IP e porto, por exemplo). Pretende-se saber também as 10 pesquisas mais comuns realizadas pelos utilizadores.

7. **(Grupos de 3 alunos) Particionamento do índice.** O índice invertido está particionado em (pelo menos) duas metades: uma contendo as palavras iniciadas pelas letras A–M e outra contendo as palavras iniciadas pelas letras N–Z. É necessário fazer "merge" de resultados de pesquisas que envolvam Barrels distintos.

5 Notificações em tempo real

De forma a que as páginas da vossa aplicação sejam atualizadas instantaneamente (server push), deverão usar WebSockets para fazer *push* de informação para o cliente assim que esteja disponível. Deverão usar WebSockets para:

- **Página de administração atualizada em tempo real.** Todos os utilizadores têm acesso a uma opção de consulta de informações gerais sobre o sistema. Esta informação será atualizada apenas quando houver alterações. Pretende-se saber o estado do sistema, designadamente a lista de Downloaders e de Barrels ativos (IP e porto, por exemplo). Pretende-se saber também as 10 pesquisas mais comuns realizadas pelos utilizadores.
- **(Grupos de 3 alunos) Particionamento do índice.** O índice invertido está particionado em (pelo menos) duas metades: uma contendo as palavras iniciadas pelas letras A–M e outra contendo as palavras iniciadas pelas letras N–Z. É necessário fazer "merge" de resultados de pesquisas que envolvam Barrels distintos. As informações apresentadas no ponto anterior devem ser complementadas com a indicação de qual das partições do índice está em qual dos Barrels ativos (IP, porto e partição, por exemplo).

6 Integração com serviço REST

Este projeto deverá ser integrado com o Hacker News. A documentação da API está disponível em `https://github.com/HackerNews/API` e permite construir as duas funcionalidades pretendidas através de REST, designadamente:

- **Indexar URLs das top stories que contenham os termos da pesquisa.** Na sequência de uma pesquisa do Googol um utilizador deve poder solicitar a indexação dos URLs das "top stories" do Hacker News que contenham (no texto) os termos da pesquisa efetuada.

ainda nao percebi o que era para fazer aqui

- **Indexar todas as "stories" de um utilizador.** Deverá ser possível pedir ao Googol para indexar os URLs de todas as "stories" de um dado utilizador do Hacker News, especificado através do seu nome de utilizador.

6.1 Relatório

Devem reservar tempo para a escrita do relatório no final do projeto, tendo em conta os passos anteriores. Devem escrever o relatório de modo a que um novo colega que se junte ao grupo possa perceber a solução criada, as decisões técnicas e possa adicionar

novos componentes ou modificar os que existem. **O relatório pode ser inteiramente escrito em Javadoc no código-fonte apresentado pelos estudantes.** Deve incluir:

- Arquitetura de software detalhadamente descrita. Deverá ser focada a estrutura de models, views, controllers, processos, threads e sockets usadas, bem como a organização do código.
- Detalhes sobre a integração do Spring Boot com o Servidor RMI da primeira meta.
- Detalhes sobre a programação de WebSockets e a sua integração com o servidor RMI.
- Detalhes sobre a integração com o serviço REST.
- Descrição dos testes feitos à plataforma.

7 Entrega do projeto

O projeto deverá ser entregue num arquivo ZIP. Esse arquivo deverá conter um ficheiro README.TXT com toda a informação necessária para instalar e executar o projeto sem a presença dos alunos. Projetos sem informações suficientes, que não compilem ou não executem corretamente **não serão avaliados**.

Dentro do ficheiro ZIP deverá também estar um Javadoc/PDF/HTML com o relatório. O relatório deve seguir a estrutura fornecida, dado que a avaliação irá incidir sobre cada um dos pontos.

Finalmente, o ficheiro ZIP deverá ter também **uma pasta com o código fonte completo do projeto**. A ausência deste elemento levará à anulação do projeto. O ficheiro ZIP deverá possibilitar executar a aplicação Web prontamente, e conter igualmente os entregáveis da meta 1 prontos a correr.

O ficheiro ZIP com o projeto deverá ser entregue na plataforma Inforestudante até ao dia **17 de maio de 2023 (21:59)**, via `https://inforestudante.uc.pt`