ucBusca: Motor de pesquisa de páginas Web

Sistemas Distribuídos 2019/20 — Meta 2 — 14 de dezembro de 2019 (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 informações pretendidas, e o número aproximado de resultados da pesquisa. Apenas os administradores podem introduzir URLs específicos para serem indexados pelo sistema. Partindo desses URLs, o sistema deve indexar iterativamente ou recursivamente todos as ligações encontradas em cada página indexada.

1 Objetivos do projeto

No final do projeto cada estudante deverá ter:

- Desenvolvido uma interface Web para a aplicação ucBusca.
- Ter integrado a interface Web com a aplicação desenvolvida na primeira meta.
- Dominado Struts2, JavaServer Pages e JavaBeans.
- Seguido a arquitetura MVC para desenvolvimento Web.
- Aplicado WebSockets para comunicar assincronamente com os clientes.
- Integrado a aplicação com serviços REST externos, usando OAuth.

2 Visão geral

Nesta segunda meta do projeto, os estudantes irão criar um *frontend* Web para a aplicação ucBusca. 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, consultar lista de pesquisas, etc. Como o aspeto interativo é muito importante na Web, o vosso projeto deverá mostrar algumas alterações em tempo real, nomeadamente ao conceder privilégios de administrador, através de notificações que envolvem a meta 1. 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 e OAuth, irão integrar a vossa aplicação com dois serviços externos: Facebook.com e Yandex.com. Deverá ser possível partilhar no Facebook as páginas com os resultados de pesquisas e traduzir para Português páginas utilizando o Yandex. A documentação relativa ao serviço Yandex está disponível em `https://tech.yandex.com/translate/doc/dg/concepts/api-overview-docpage/` e a documentação do serviço do Facebook pode ser acedida em `https://developers.facebook.com/docs/graph-api/reference/`.

3 Arquitetura

A Figura 1 mostra a arquitetura geral do projeto. Os elementos a amarelo referem-se à segunda meta do projeto, enquanto os outros se referem à primeira meta. 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 (Apache Tomcat) 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. Isto inclui notificações e alterações de valores em tempo real.

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 Struts2:

1. **Registar pessoas.** Os utilizadores devem poder registar-se, sendo que o acesso à aplicação deve estar protegido com username e password. Deverá guardar toda a informação pessoal que considere necessária bem como uma password (código de acesso). Para simplificar, considere que o primeiro utilizador a registar-se tem automaticamente privilégios de administrador em todo o sistema.

2. **Indexar novo URL.** Um administrador deve poder introduzir manualmente um URL para ser indexado. Esse URL será então visitado pelo indexador automático (*Web crawler*) e as palavras que forem encontradas no texto serão acrescentadas ao índice invertido.

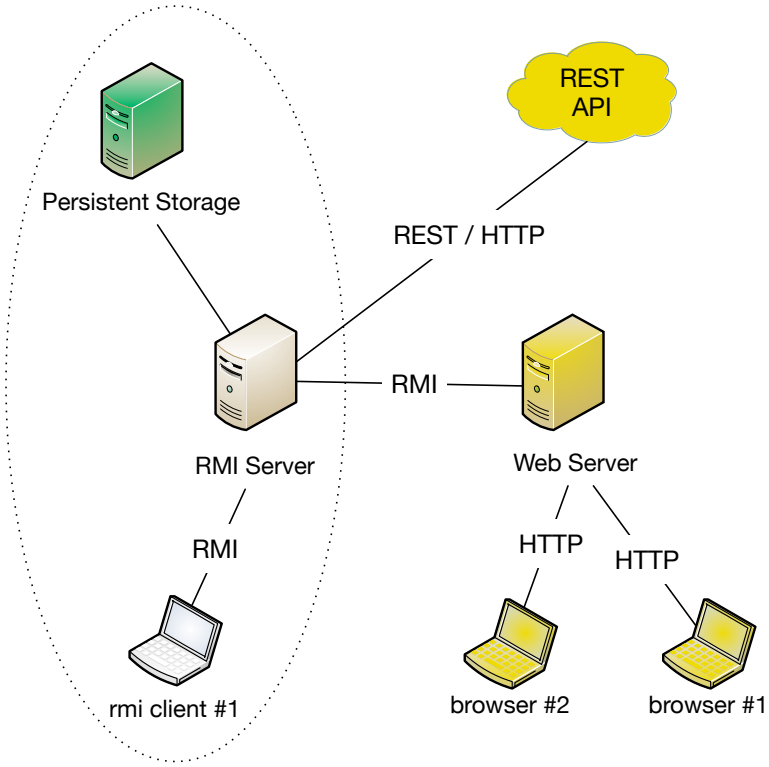

Fig. 1: Arquitetura do projeto

3. **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.

4. **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. Deve ser apresentado o número total aproximado de resultados da pesquisa. Esta funcionalidade é a única que deve estar disponível para utilizadores que não tenham login efetuado (utilizadores anónimos).

5. **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.

6. **Consultar lista de páginas com ligação para uma página específica.** Aos uti-

lizadores que tenham registo e login efetuado é possível saber, para cada página, todas as ligações conhecidas que apontem para essa página. Esta funcionalidade pode estar associada à funcionalidade de pesquisa.

7. **Consultar pesquisas realizadas anteriormente.** Os utilizadores que tenham registo e login efetuado podem consultar todas as pesquisas que os próprios tenham feito anteriormente.

8. **Dar privilégios de administrador a um utilizador.** Aos utilizadores que tenham privilégios de *administrador* é possível conceder esses mesmos privilégios a outros utilizadores. Para tal, escolhem o utilizador que pretendem promover a administrador e esse utilizador passa a poder alterar informações no sistema.

9. **Entrega posterior de notificações a utilizadores desligados.** Qualquer utilizador que devesse ter recebido uma notificação imediata, mas não se encontrasse ligado à aplicação (offline), recebe a notificação assim que se ligar a próxima vez.

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:

- **Notificação imediata de privilégios de administrador.** Quando um utilizador é promovido a administrador deve receber imediatamente (em tempo real) essa informação se estiver ligado à aplicação, sem precisar de realizar nenhuma operação.

- **Página de administração atualizada em tempo real.** Os administradores têm acesso a uma opção de consulta de informações gerais sobre o sistema. Esta informação será atualizada apenas quando houver atualizações. Pretende-se saber as 10 páginas mais importantes e as 10 pesquisas mais comuns realizadas pelos utilizadores.

- **Atualização imediata da lista de servidores multicast ativos.** Na funcionalidade anterior, os administradores têm acesso a uma opção de consulta de informações gerais sobre o sistema. Para além dessas informações, devem ter também acesso à lista de servidores multicast da meta 1 que estejam ativos (IP e porto).

6 Integração com serviço REST

Este projeto deverá ser integrado com o Facebook e com o Yandex. O Facebook será usado para partilhar pesquisas efetuadas com o ucBusca, bem como fornecer uma alternativa ao login por username e password. Para usar a API do Facebook é necessário usar autenticação com OAuth. Para tratar deste processo devem usar a biblioteca Scribejava. Não serão aceites bibliotecas que façam a integração em Java com o Facebook.

O Yandex será utilizado para traduzir o texto apresentado nos resultados de pesquisas e para detetar qual a língua de páginas.

As funcionalidades a implementar com recurso a REST são:

- **Associar conta ao Facebook.** Qualquer utilizador com login efetuado poderá associar a sua conta no ucBusca à sua conta do Facebook. Este passo permitirá depois fazer login com a conta do Facebook e aceder à informação que lá se encontre.
- **Partilha de um resultado de pesquisa.** Ao realizar uma pesquisa (funcionalidade 4) deverá ser possível a um utilizador partilhar no Facebook a página com os resultados da pesquisa. Esta seria uma forma de atrair utilizadores ao ucBusca.
- **Língua original de cada página mostrada nas pesquisas.** A página com os resultados de pesquisas deve mostrar, junto a cada URL, a língua na qual está escrito. A API do Yandex deverá ser utilizada para detetar a língua de cada URL indexado que apareça nos resultados de uma pesquisa.
- **Traduzir título e citação das páginas para Português.** A página com os resultados de pesquisas (funcionalidade 4) deverá apresentar, para cada resultado, o título, uma citação curta de texto e o URL completo. Um utilizador deverá ter uma opção para obter uma versão traduzida da pesquisa, na qual os títulos e as citações são mostrados em Português (através da API do Yandex).
- **Registo com a conta do Facebook.** Um utilizador que ainda não tenha feito registo, deverá poder fazê-lo com o Facebook, entrando automaticamente na sua conta ucBusca sem ter de inserir username ou password. Isto permite a alguém experimentar o ucBusca sem criar uma conta especificamente na aplicação.

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 Struts2 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. Também no ficheiro ZIP deverá existir um ficheiro WAR com a aplicação Web pronta a executar, bem como os entregáveis da meta 1 prontos a correr.

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 com o projeto deverá ser entregue na plataforma Inforestudante até ao dia **14 de dezembro de 2019 (21:59)**, via `https://inforestudante.uc.pt`