Relatório do Problema 3: Motor de Busca FeiraGugou

Anésio Sousa dos Santos Neto

Departamento de Tecnologia
Universidade Estadual de Feira de Santana (UEFS)
Av. Transnordestina, s/n, Novo Horizonte – 44036-900
Feira de Santana – BA – Brasil

anesios98@gmail.com

Abstract. *This report describes all the steps for the development of a search engine similar to Google, using the MVC architecture pattern and a Binary Search Balanced Tree for optimization of the processes of search, insertion and removal of data. The entire project was developed using the git versioning tool.*

Resumo. *Este relatório descreve todos os passos para o desenvolvimento de um motor de busca similar ao Google, utilizando o padrão de arquitetura MVC junto à Árvore Binária de Busca Balanceada para otimização dos processos de pesquisa, inserção e remoção de dados. Todo o projeto foi desenvolvido utilizando a ferramenta de versionamento git.*

1. Introdução

É de conhecimento geral que no quesito de busca de informações o Google é líder absoluto. Frases como "Tenho cara de Google?" ou "Confesso que fiz mais perguntas ao Google do que a Deus" se tornaram comuns devido a grande popularidade do mesmo. Fazer o motor de busca do Google não foi brincadeira. Para chegar onde está hoje foram anos de trabalho duro e desempenho de muitas pessoas que são sinônimo de qualidade. Para sentirmos um gostinho do que é um motor de busca, foi nos pedido para fazer um pequeno motor de busca onde a área de atuação dele seriam arquivos de texto armazenados em uma pasta do computador.

2. Fundamentação teórica

Para criar o sistema foi necessário reunir algumas informações vitais para o cumprimento do que foi proposto, escolhendo formas e ferramentas para um melhor desenvolvimento do produto. Ferramentas como: linguagem a ser utilizada, padrões de projeto a serem utilizados, estruturas de dados que seguissem a complexidade algorítmica pedida e algoritmos de ordenação de dados.

2.1. Linguagem de programação

A linguagem escolhida para a criação do sistema foi o Java. Mesmo anos de existência, Java é uma linguagem bastante atual. É uma grande referência quando tratado de linguagens orientadas a objetos e é comumente utilizada em pequenos e grandes projetos pois tem uma bibliotecas ricas, tem grande portabilidade, tem uma grande

comunidade, e etc.

2.2. Padrões de projeto

O padrão arquitetural escolhido para o desenvolvimento do sistema foi o MVC. A divisão em camadas que esse padrão trás, faz com que o sistema se torne algo mais fácil de serem adicionadas funcionalidades, ter uma melhor manutenção, e se for o caso, o torna mais simples de ser refatorado.

2.3. Manipulação de arquivos

Para serem a páginas da web, foram escolhidos arquivos de texto, onde numa pesquisa comum do usuário, são retornados resultados da pesquisa nesses arquivos de texto armazenados em um repositório local.

2.4. Ordenação de dados

Ordenar ou classificar alguma coisa é organizá-la de uma maneira bem definida. Ao ordenar números por exemplo, é possível ordená-los pela ordem natural, ordem inversa, pares, ímpares e uma gama extensa de possibilidades. Em uma exibição de resultados em uma pesquisa no site Google, esses resultados são exibidos de uma forma decrescente, onde páginas que contém uma maior relevância são mostradas mais acima das menos relevantes. Como relevância é um termo relativo, a relevância do software em questão é dada pela frequência do termo pesquisado nas páginas. Sendo assim, uma página com um grande número de ocorrências do termo é exibida mais acima das outras.

2.5. Estruturas de dados

Um bom uso de estruturas de dados transforma drasticamente a experiência do usuário com um software. Buscando rapidez e eficiência nas pesquisas e nos resultados das pesquisas, foi utilizada uma Árvore Binária de Busca Balanceada (AVL). Sua incrível eficácia em retornar um elemento buscado com rapidez, e sua organização estruturada de elementos, a torna mais do que capaz de suprir as necessidades do sistema.

2.6. Ferramentas extras

Expressões regulares foram utilizadas identificar um padrão e separar palavras de caracteres de texto.Para a criação do software foi utilizado Linux Mint 18.3 como sistema operacional, e o ambiente integrado de desenvolvimento Netbeans IDE 8.2.

3. Metodologia

A existência atualmente de muitos motores de busca como o Google, Yahoo! e o bing, criou uma base sólida de passos a serem feitos para a criação do software em questão. As estratégias utilizadas por essas grandes empresas na obtenção de dados para mostrar em seus resultados de pesquisas, utilizam um alto nível de profissionalismo, abstração e excelente uso de estruturas de dados por meio de seus engenheiros. Tomando essa mesma visão, foi buscado apresentar um software que cumprisse bem seus requisitos.

3.1. Padrão MVC

Usando o padrão MVC, O sistema foi subdividido em 4 pacotes, sendo eles:

Model : Onde ficam as regras de negócio do sistema. As classes responsáveis por gerir o funcionamento principal. Nesse pacote ficaram as classes Dados, Página, e Palavra. A classe Página ficou responsável por ser um reflexo dos arquivos de texto do repositório. Cada objeto Página representa um arquivo de texto. Páginas armazenam os títulos dos arquivos de texto, as vezes que foi pedido para abrir esses arquivos de texto (acessos às páginas), e a informação da última vez que esse arquivo de texto foi modificado. A classe Palavra ficou responsável por armazenar as palavras chave buscadas pelos usuários. Cada objeto Página armazena a palavra chave buscada, quantas vezes essa palavra foi buscada e uma lista de objetos Dados.A classe Dados ficou responsável por fazer uma junção entre as Palavras e as Páginas. Ela foi necessária pois várias palavras tinham tinham ocorrências em vários arquivos. Armazenar essas informações nas próprias Páginas e nas próprias Palavras se mostrou inviável. Cada objeto Dados na lista de dados de uma Palavra armazena o título da página que tal Palavra ocorre, e uma quantidade de vezes que tal palavra ocorre em tal página.

View: Onde ficam as classes responsáveis por fazer a interação homem-máquina. Neste pacote ficou a classe ConsoleView, que não é nada além de um ambiente de testes. A intenção era ser feito um ambiente completo gráfico para os usuários mas por contratempos não foi possível.

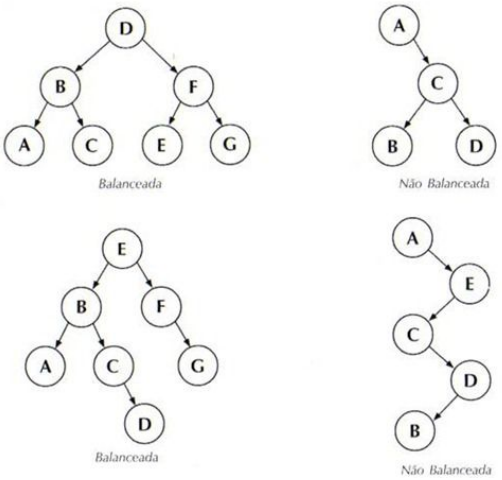
Controller: Onde ficam as classes responsáveis pelo fluxo da aplicação. Nesse pacote ficaram 3 classes responsável pelo gerenciamento do fluxo: GerenciadorDePesquisa, GerenciadorDePaginas e GerenciadorDeArvore. No gerenciador de pesquisa ficam os métodos responsáveis por fazer com que a pesquisa seja feita corretamente. Lá é instanciado um gerenciador de páginas, que por sua vez verifica a integridade das páginas para que a árvore de palavras e as páginas estejam em comum acordo. No gerenciador de páginas é instanciado o controlador de árvore que por sua vez mantém o controle da árvore de palavras.

Util: Onde ficam as classes responsáveis por serem ferramentas públicas de trabalho. Lá ficam a classe responsável por gerar árvores, a classe MergeSort, que é uma classe que possibilita ordenação de ArrayLists sem e com Comparator, e a classe Arquivos, que ajuda a pegar dados do repositório e passar para a lista de páginas.

3.2. Árvore AVL

Pela eficiência nas operações de busca, inserção, remoção (log n) e balanceamento, foi escolhida a Árvore Binária de Busca Balanceada. Visto que em sistemas de pesquisa de dados, é necessário uma resposta rápida ao usuário. Sua principal diferença entre árvores binárias de busca comuns é a questão do balanceamento, onde em um pior caso de serem só inseridos sequências ordenadas de elementos, ela consegue organizá-los de uma maneira que ainda fique muito rápida as ações de remoção, inserção e busca na árvore (Figura 1).

Figura 1 : Comparação entre árvore AVL balanceada e não balanceada



Fonte: Eduardo Spaki - Árvores pt III: Teoria, Introdução, Binárias e AVL

4. Resultados e discussão

O resultado do projeto é um motor de busca similar a qualquer outro, mas com um plus, que é o ranking das palavras mais e menos procuradas e das páginas mais e menos acessadas Figura 2. Só é possível pesquisar uma palavra por vez. O sistema de verificação de integridade dos arquivos está incompleto, pois a falta de planejamento acarretou em bugs e horas de tempo gastas para arrumá-los sem sucesso. Não irei falar muito sobre a interface, pois a interface planejada para o sistema não pôde ser completada por causa dos contratempos. A GUI até onde foi possível ser feita, se encontra na pasta “extras” do projeto.

Figura 2. Menu principal do programa

```
+=============================================================================+
|                                FeiraGugou                                   |
+=============================================================================+
| Realizar Busca.....................................................(01) |
| Top-K..............................................................(02) |
+=============================================================================+
| Sair...............................................................(00) |
+=============================================================================+
>
```

Fonte: Próprio Autor

Os requisitos principais do projeto foram feitos, mas problemas como lentidão na pesquisa mesmo com a palavra já na árvore se tornaram difíceis de se resolver na conjuntura atual de verificação de arquivos.

5. Conclusão

Depois de todo o processo de desenvolvimento do sistema, é com mágoa que tenho que falar mal do meu próprio projeto. Gostaria mesmo de ter feito a interface gráfica,

gostaria mesmo de ter feito a verificação usando thread, mas contratempos pessoais e no projeto acabaram não deixando. Mesmo sabendo que a grande maioria do projeto foi feita, é triste entregar algo incompleto por falta de tempo. Levo comigo a experiência de fazer algo assim, para que não o faça novamente.

Referências

SIERRA, K; BATES, B. **Use a cabeça!: Java.** 2. ed. Alta Books

BAPTISTELLA, J, Adriano. **Abordando a arquitetura MVC, e Design Patterns: Observer, Composite, Strategy.** Disponível Em: <http://www.linhadecodigo.com.br/artigo/2367/abordando-a-arquitetura-mvc-e-design-patterns-observer-composite-strategy.aspx> Acesso Em: 28 de maio de 2018.

FISET, William. **AVL tree source code.** Disponível em: <https://www.youtube.com/watch?v=tqFZzXkbbGY> Acesso Em: 17 junho de 2018.

SIERRA, K; BATES, B. **Use a cabeça! Padrões de Projeto.** 2. ed. Alta Books