

**PROGRAMAÇÃO AVANÇADA**

**Joanã Costa 180 221 109**

**Daniel Cordeiro 190 221 101**

**WebCrawler**



Índice

[1. Introdução 3](#_Toc30364297)

[2. Tipos Abstratos de Dados 3](#_Toc30364298)

[3. Diagrama de Classes 4](#_Toc30364299)

[4. Documentação de Classes (Javadoc) 5](#_Toc30364300)

[5. Padrões de Software 6](#_Toc30364301)

[MVC 6](#_Toc30364302)

[6. Refactoring 7](#_Toc30364303)

[7. Conclusão 8](#_Toc30364304)

# Introdução

Este projeto tem como objetivo a utilização da TAD ***Graph*** e ***Queue*** na resolução do problema de percorrer páginas *web* e extrair demais páginas associadas. Sendo assim será desenvolvido um ***WebCrawler***que fará essa tarefa através dos algoritmos ***breadth-fisrt*** e eventualmente o algoritmo *Djisktra*.

# Tipos Abstratos de Dados

Para este projeto usamos alguns tipos para facilitar a manipulação dos dados.

Esses são os tipos de dados usados:

* ***HashMap*** – Usados para guardar os vértices e as arestas na classe ***MyDigraph***;
* ***ArrayLists*** – Usados para guardar vários tipos de dados ao longo do código;
* ***Queue*** – Usado para armazenar em memória os objetos das páginas Web para que sejam retiradas por ordem e processadas as mesmas;

Tipos abstratos de dados implementados:

* ***MyDigraph*** – Tipo de dados que representa um grafo, usado para representar as páginas web que tem ligações com outras.
* ***MyVertex*** – Tipo de dados que representa um vértice do grafo;
* ***MyEdge*** – Tipo de dados que representa uma aresta do grafo;

# Diagrama de Classes

Na figura que se segue estão representadas as classes utilizadas na 1º fase do desenvolvimento da aplicação do ***WebCrawler*** e os acoplamentos entre elas.

Uma imagem com texto

Descrição gerada automaticamente

Uma imagem com texto

Descrição gerada automaticamente

# Documentação de Classes (Javadoc)

# Padrões de Software

Durante o decorrer da UC foram lecionados alguns padrões de software dos quais usados foram:

## MVC e Observer

A estrutura do nosso projeto está baseada neste padrão devido a sua robustez e distribuição de responsabilidade entre classes, cada classe é responsável pelo que lhe foi imputado. Sendo assim temos três packages no nosso projeto, **Model-Views-Controller**, está representado numa figura abaixo a estrutura no ***NetBeans***.

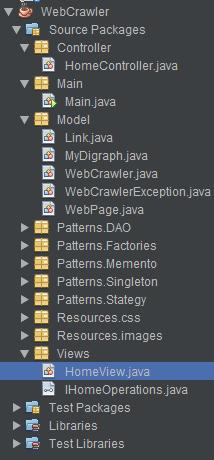


Figura 1 - Estrutura NetBeans

## DAO (Data Access Object)

Como no nosso projeto precisaríamos exportar ficheiros de texto e em JSON, tivemos que implementar este padrão para esses dois tipos de ficheiro. Este está representado na classe ***WebCrawlerFile*** e ***WebCrawlerJson*** onde é usado o modelo (classe ***WebCrawler***).



Figura 2 - Classes dos tipos de ficheiros

## Strategy

O padrão strategy é usado para alternar entre os critérios de paragem no modo iterativo e nos modos automáticos.

No modo automático temos três critérios de paragem sendo eles:

* Páginas visitadas – Termina assim que n páginas forem visitadas;
* Profundidade – Termina assim que todas as páginas de que distam M links da página inicial;
* Uma página que tem mais do que n links, não será mais expandida;

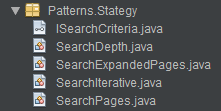


Figura 3 - Representação das Estratégias

## Simple Factory

Para este padrão usamos para criar objetos pertencente ao padrão MVC e os tipos de ficheiro juntamente com o DAO. As classes usadas são a ***Main, WebCrawler e WePage***.



Figura 4 - Classe Factory

## Singleton

Para este padrão, usamos para fazer os ***logs*** por ficheiro de texto do programa. Este é chamado em quase todos as classes do programa para notificar erros que ocorram durante a execução do programa e para escrever as páginas web que foram inseridos durante a pesquisa das mesmas.

Classes Usadas:

* WebCrawler
* WebPage
* WebCrawlerFile
* WebCrawlerJson
* SearchPages
* SearchIterative
* SearchDepth



Figura 5 - Classe Singlenton

## Memento

Apesar de não termos conseguido implementar de forma perfeita no nosso projeto, este também está presente na classe principal de modelo do ***WebCrawler***, onde o objetivo era ir guardando os estados do grafo quando a página era expandida e permitir fazer o undo dessa ação se o utilizador quisesse.

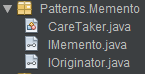


Figura 6 - Implementação Memento

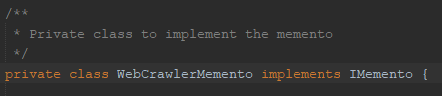


Figura 7 - Implementação da inner class no modelo

# Refactoring

# Conclusão

**Rafael Trindade - 160210033**

**Vítor Gomes - 160210050**