Universidade da Beira Interior
Departamento de Informática

JC1- Cristal News - Visualização e Caracterização de Notícias Web

Elaborado por:

Luís Carlos Durão Mata de Oliveira

Orientador:

Professor Doutor João Paulo Cordeiro

9 de Julho de 2018

Agradecimentos

Quero agradecer desde já ao meu orientador, Professor Doutor João Paulo Cordeiro, por todo o apoio que me deu ao longo da realização do projecto, bem como a oportunidade que tive em alargar o meu conhecimento na área da interação humana com o computador. Gostaria também de agradecer a todos os meus colegas que me apoiaram e encorajaram. Por fim gostava de agradecer à minha família, os meus pais, José Manuel Oliveira e Maria de Lurdes Oliveira e à minha irmã, Ana Isabel Oliveira, pela educação e apoio que me deram ao longo deste caminho.

Conteúdo

Conteúdo **iii**

Lista de Figuras **v**

Lista de Tabelas **vii**

1 Introdução **1**
1.1 Enquadramento 1
1.2 Motivação 1
1.3 Objetivos 2
1.4 Organização do Documento 2

2 Estado da Arte **3**
2.1 Introdução 3
2.2 Modelo pirâmide invertida 3

3 Tecnologias e Ferramentas Utilizadas **5**
3.1 Introdução 5
3.2 Recursos Utilizados 5
3.2.1 NetBeans 5
3.2.2 SceneBuilder 6
3.2.3 JavaFx 6
3.2.4 JSOUP 8
3.2.5 GitHub 10
3.3 Conclusões 10

4 Implementação e Testes **11**
4.1 Introdução 11
4.2 Implementação 11
4.2.1 HTML dinâmico 11
4.2.2 DestkNew 12

4.2.3 getParagraph 14
4.3 Conclusões 15

5 Conclusões e Trabalho Futuro **17**
5.1 Introdução 17
5.2 Conclusões Principais 17
5.3 Trabalho Futuro 17

Bibliografia **19**

Lista de Figuras

2.1 Pirâmide invertida. 4

3.1 *Scene Builder*. 6
3.2 Diagrama de interacções do Model-view-controller (MVC). 7
3.3 Imagem parcial da aplicação. 8
3.4 *Tag img* com imagem da notícia. 9

4.1 Exemplo de notícia destacada. 14

Lista de Tabelas

4.1 Exemplo de interseção de dois léxicos. 13

Acrónimos

HTML Hyper Text Markup Language

API Application Programming Interface

MVC Model-view-controller

Capítulo 1

Introdução

Este capítulo é dedicado à apresentação do tema do projecto bem como o porque da sua escolha, vão ser apresentados os seus objectivos e estruturação deste documento.

1.1 Enquadramento

Atualmente o jornalismo evolui no sentido em que cada vez é mais lógico e normal investir em notícias online, com isto todos os dias são colocadas enormes quantidades de notícias novas na Internet, ficando desta forma disponíveis para milhões de pessoas.

Devido a este grande fluxo de notícias só temos um perceção parcial da informação noticiosa que circula nos meios eletrónicos. Assim sendo um utilizador algumas tem muitas dificuldades em encontrar as notícias em que está interessado com a facilidade que desejaria. Ele só conseguirá encontrar noticias de maior importância providenciadas pelos organizações mais relevantes.

1.2 Motivação

Este projeto visa a criação de uma aplicação que a partir de uma grande e variedade de notícias produz uma vista clara para os tópicos escolhidos por um utilizador. Através de uma interface atualizada em tempo real e com o maior número de notícias possíveis, de acordo com o tópico escolhido, permite ao utilizador ter uma visão abrangente sobre os temas, reduzindo o tempo de pesquisa tornando mas simples a procura de notícias.

1.3 Objetivos

O objectivo deste projecto passa por testar, estudar e experimentar medidas de *web srcaping* de forma a sintetizar o número de notícias presentes na *web*. Através de uma aplicação o utilizador terá acesso a noticias separadas por tópicos, em cada tópico será possível realçar notícias consoante um léxico de palavras escolhido pelo utilizador.

1.4 Organização do Documento

De modo a refletir o trabalho que foi feito, este documento encontra-se estruturado da seguinte forma:

1. O primeiro capítulo – **Introdução** – apresenta o projeto, a motivação para a sua escolha, o enquadramento para o mesmo, os seus objetivos e a respetiva organização do documento.

2. O segundo capítulo – **Tecnologias Utilizadas** – descreve os conceitos mais importantes no âmbito deste projeto, bem como as tecnologias utilizadas durante do desenvolvimento da aplicação.

3. O terceiro capítulo – **Tecnologia e Ferramentas Utilizadas** – descreve as tecnologias utilizadas durante o desenvolvimento do projecto

4. o quarto capítulo – **Implementação e Testes** – apresenta e discute os resultados obtidos através do uso da aplicação desenvolvida

5. O quinto capítulo – **Conclusões e Trabalho Futuro** – descreve as conclusões obtidas pelo autor, bem como algumas sugestões de desenvolvimento do projecto no futuro.

Capítulo 2

Estado da Arte

2.1 Introdução

O Capítulo 2 vai ser apresentado um modelo de redação de notícias, que se mostrou bastante útil neste projeto.

2.2 Modelo pirâmide invertida

A notícia não é mais do que texto jornalístico relativo a um conteúdo factual e relata acontecimentos de interesse geral com a menor subjetividade possível. O texto da notícia deve ser escrito com clareza, simplicidade e exatidão, assim como o bom uso do português e do cumprimento da gramática. Uma notícia presente na *Internet*, requer um tratamento diferente das demais, estas combinam texto som e imagem.

O modelo pirâmide invertida, tem uma função interessante no que diz respeito à redação de notícias, consiste em apresentar primeiro as informações mais importantes e na sequência da notícia apresentar as informações menos importantes. Isto permite ao leitor ter uma noção do que será apresentado ao longo da notícia, só através do primeiro parágrafo.

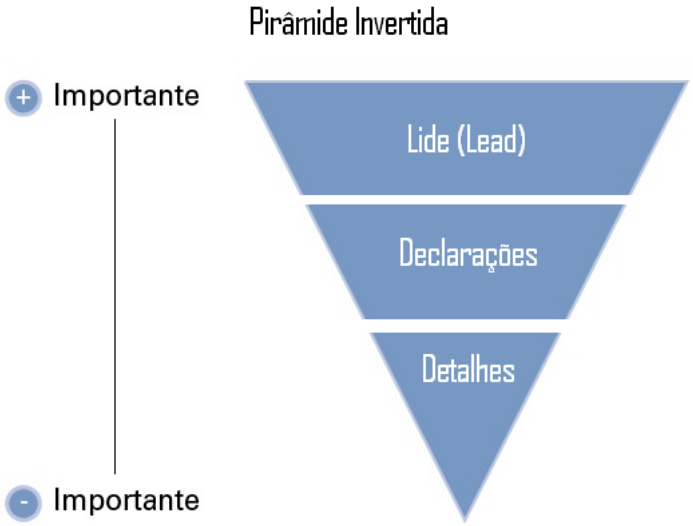

Figura 2.1: Pirâmide invertida.

Na Figura 2.1 observamos que a pirâmide começa com o *Lide(Lead)*, este não é mais do que a introdução da notícia. É onde se situa o leitor com os factos que aconteceram, deve conter respostas às seguintes perguntas: 'O que?', "Quando?", "Quem?", "Como?", "Onde?"e principalmente "Porque?". A secção *Declarações* tem como objetivo desenvolver a notícia, suportando-a com informações, entrevistas ou referências. À medida que o notícia se desenrola os *Detalhes* cada vez menos importantes por isso têm a posição mais baixa para pirâmide.

Com base neste modelo utiliza-se o primeiro parágrafo de uma notícia para ser mostrado na aplicação desenvolvida.

Capítulo 3

Tecnologias e Ferramentas Utilizadas

3.1 Introdução

O capítulo 3 é dedicado à identificação das ferramentas e tecnologias usadas para desenvolver este projeto. Foram usadas algumas das seguintes ferramentas: *JavaFx* para desenvolver a interface gráfica da aplicação, a biblioteca *JSOUP* uma ferramenta que permite analisar um documento *HTML* e *GitHub* para controlo de versões.

3.2 Recursos Utilizados

3.2.1 NetBeans

O *NetBBeans* é um ambiente de desenvolvimento de código, onde é possível desenvolver uma grande variedade de linguagens, dentro das quais *Java*, usada neste projeto. Auxilia programadores a escrever, compilar, no *debug* de código e até mesmo a instalar aplicações.

A sua estrutura está feita de forma a simplificar o desenvolvimento e aumentar a produtividade, uma vez que reúne numa única aplicação todas as funcionalidades descritas anteriormente.

O *NetBeans* fornece uma base sólida para a criação de projetos, possui um grande conjunto de bibliotecas e Application Programming Interface (API), além de uma documentação vasta. Isto faz com que o utilizador escreva o seu código de uma forma mais rápida.

Optei por escolher o *NetBeans* porque já tinha experiência anterior na sua utilização e pareceu a mais adequada para usar em conjunto com o *SceneBuiler*.

3.2.2 SceneBuilder

O *SceneBuilder* é uma plataforma de desenvolvimento de ambientes gráficos, que permite criar rapidamente uma aplicação *JavaFx*, sem recurso a código.

Os utilizadores podem arrastar componentes, modificar as suas propriedades, aplicar estilos e o código *FXML* é gerado automaticamente. Esse ficheiro *FXML* é combinado com código *Java*.

Permite moldar facilmente o ambiente gráfico da aplicação e adequa-lo às nossas necessidades. O código *FXML* é gerado automaticamente, separadamente da interface lógica da programação. É possível abrir o ficheiro *FXML* e editar o seu código.

A qualquer momento da criação do projeto, é possível pré-visualizar a aplicação durante o seu desenvolvimento.

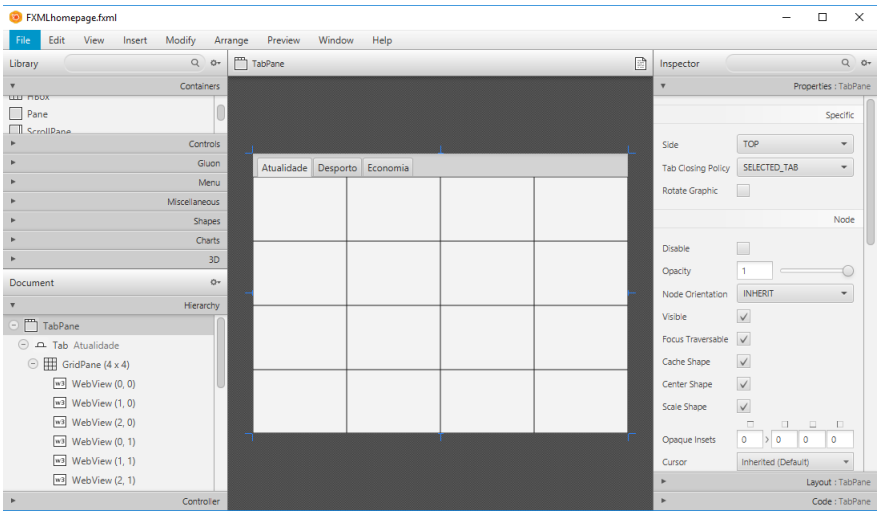

Figura 3.1: *Scene Builder*.

Na figura 3.1 é possível ver uma interface que foi desenvolvida no projeto, permitiu criar de forma rápida uma página que contem muitos elementos. Foi usado para desenvolver o ambiente gráfico do projeto, fazendo uma ligação rápida com o código *Java*.

3.2.3 JavaFx

JavaFX é uma plataforma que permite desenvolver aplicações com interface gráfica com base na programação por eventos. Os *eventos* tratam-se nada mais do que a captura de ações desencadeadas pelo utilizador, tais como, carregar num

botão ou numa imagem. Quando isso acontece certos eventos são desencadeados, originando uma resposta.

O *JavaFx* interage com o utilizador de acordo com o modelo MVC, este consiste em dividir uma aplicação em três partes interligadas. Este modelo é usado para separar as representações internas da informação da forma como a informação é apresentada e aceite pelo utilizador.

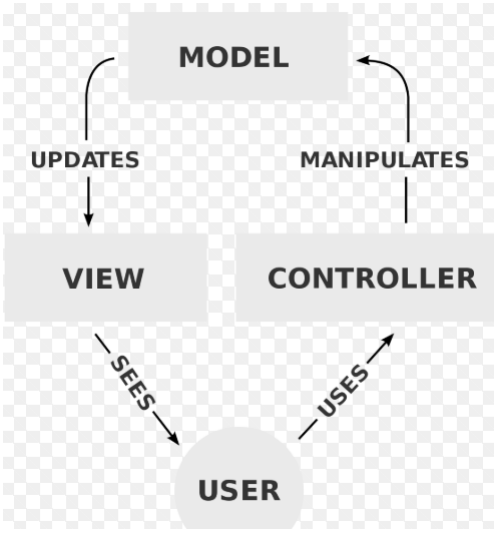

Figura 3.2: Diagrama de interacções do MVC.

Na figura 3.2 está representado o diagrama de interações do modelo MVC. Observamos que o *Model* é a componente central, é independente do utilizador e gere os dados, a lógica e regras da aplicação. A *View* pode ser qualquer representação da informação, tais como um gráfico ou um diagrama. A terceira parte *Controller* aceita o *input* e converte-o em comandos para serem interpretados pelo *Model*

Em conjunto com o *SceneBuilder*, o *JavaFx* demonstrou ser uma excelente ferramenta para o desenvolvimento de aplicações com interface gráfica, em conjunto estas duas ferramentas foram usadas para desenvolver a seguinte interface gráfica:

Figura 3.3: Imagem parcial da aplicação.

A interface gráfica é particularmente simples de usar, tem como objetivo principal mostrar notícias e destacar aquelas que o utilizador considera mais importantes, através de um realçar da cor de fundo da própria notícia. Através do uso de separadores é possível escolher diferentes temas das notícias.

É possível atualizar as notícias carregando no botão *"R"* e também são automaticamente atualizadas a cada meia hora, isto faz com que a aplicação esteja sempre atualizada.

3.2.4 JSOUP

Jsoup é uma biblioteca *Java* que permite analisar, manipular e extrair informações guardadas num documento Hyper Text Markup Language (HTML). No âmbito

deste projeto revelou-se muito importante, uma vez que através do *Jsoup* que consegui manipular *websites*, extraindo o seu código HTML e a partir daí usar as informações a meu gosto.

Foram usadas várias funcionalidades do *Jsoup*, tais como objetos do tipo *Document*, que representam o código HTML de uma página e objetos do tipo *Element*, estes representam *tags* HTML e é a partir delas que navegamos pelo documento.

```
try {
    Document document = Jsoup.connect(url).get();

    Elements locator = document.select("article[data-kpi] >
        figure > a > img");

    for (Element element : locator) {
        img_links.add(element.attr("src"));
    }
    } catch (IOException e) {
        e.printStackTrace();
}
```

Listing 3.1: Selecionar imagens de *website*

Na Listing 3.1 podemos observar em primeiro lugar que é feita uma conceção com o site, isto faz com que o código HTML seja carregado para o objeto *document*.

Pode-se observar o método *select*, este permite encontrar elementos de acordo com uma *query* feita ao documento. O *select* presente na imagem dá origem a todos as *tags img* que contêm a imagem de uma notícia. Considerando o código HTML está estruturado como uma árvore, primeiro procura-se por uma *tag article* que contenha um atributo *data-kpi*, a partir daqui procuram-se descendentes diretos (filhos), até encontrarmos a *img* que pretendemos. A imagem seguinte revela exatamente o que foi descrito anteriormente.

Figura 3.4: *Tag img* com imagem da notícia.

Este é um exemplo da utilização do *Jsoup* no projeto, existem outros semelhantes que retiram o título das notícias ou o primeiro parágrafo da notícia, a técnica usado é em todo semelhante à do exemplo representado anteriormente.

3.2.5 GitHub

O *GitHub* é um sistema de controlo de versões de código, através do qual é possível realizar uma gestão mais eficaz do projeto sem correr o risco de apagar versões anteriores. Foi útil para este projeto porque permitiu guardar versões anteriores, muitas vezes ao implementar funcionalidades novas era possível testar versões anteriores em separado. O repositório deste projecto pode ser consultado em https://github.com/luisoliveira22/cristalNews.git.

3.3 Conclusões

Neste capítulo foram descritas e caracterizadas detalhadamente as ferramentas e tecnologias usadas neste projeto, portanto, explicado o porquê da sua utilização.

Capítulo 4

Implementação e Testes

4.1 Introdução

O capítulo 4 e dedicado á apresentação de algumas funções implementadas no âmbito do projeto e também alguns testes efetuados.

4.2 Implementação

4.2.1 HTML dinâmico

Durante o desenvolvimento deste projeto, foi usada a construção dinâmica de um ficheiro HTML, que diz respeito a uma notícia.

```
String html
    = "<html>"
    + "    <head>"
    + "        <style>"
    + "            p { text-align: justify;}"
    + "        </style>"
    + "    </head>"
    + "    <body bgcolor=" + "\"" + bgcolor + "\">"
    + "        <img src=" + "\"" + imagensAtualidade.get(i) + "\""
        + "align=" + img_align + " width=\"150\" height=\"130\"
       />"
    + "        <p><b>" + titulosAtualidade.get(i) + "</b><br>"
    + paragrafosActualidade.get(i)
    + "        </p>"
    + "    </body>"
    + "</html>";
webviewActualidade[i].getEngine().loadContent(html);
```

Listing 4.1: Exemplo de construção de HTML dinâmico

Neste caso foram usados métodos de *Jsoup* para extrair a informação pretendida de uma página. Essas informações são armazenadas em *ArrayList<String>*, e é a partir delas que se constrói o HTML. Estão presentes três *ArrayList* diferentes, *imagensAtualidade, titulosAtualidade e paragrafosAtualidade*. Cada uma armazena informações em *Strings* referentes às imagens, aos títulos e aos parágrafos, das noticias.

No fim de a *string html* estar finalizada, necessitamos de mostrar o seu conteúdo. Para isso é usado um objeto do tipo *WebView*, este permite exibir uma página *web*, quer através do *link* direto para o mesmo ou através de uma *string*, o último é usado neste projeto. Através dos métodos *getEngine().loadContent(html)* é possível carregar o conteúdo da string para o *webview*, neste caso se as dimensões da página forem superiores às dimensões do *webview*, este cria um *scroll* automático. Esta é uma propriedade bastante interessante pois em grande parte dos casos as notícias excedem o tamanho do *webview*.

4.2.2 DestkNew

Esta função determina se uma notícia vai ser ou não destacada, alterando a cor de fundo da mesma. Tem como objetivo realçar uma notícia aos olhos do utilizador.

```
private Boolean destakNew(String texto_noticia)
HashSet<String> textNot = new HashSet<>();
String[] palavras = texto_noticia.trim().split("[ ,;:.!?\n]+");
for (String palavra : palavras) {
    textNot.add(palavra);
}

int lexSize = lexicoRelevante.size();
textNot.retainAll(lexicoRelevante);
double conjuction = (double) textNot.size() / lexSize;

if (conjuction > 0.35) {
    return true;
} else {
    return false;
}
```

Listing 4.2: Código usado para destacar notícia

Para isso foi usado um léxico de palavras escolhido pelo utilizador, esse léxico é comparado com o texto da notícia e no final só é destacada se a interseção dos dois conjuntos for superior a 35%.

Assim sendo, são usados objetos do tipo *HashSet<String>*, este cria uma estrutura de *string* que usa tabelas de *hash* para armazenar os dados. Não permite objetos duplicados e e não segue a ordem de inserção dos dados, cada *string* é

identificada com um número único.

Na Listing 4.2 visível o *lexicoRelevante*, um *HashSet* que carrega as palavras de um ficheiro. Este ficheiro é manipulado pelo utilizador e deve conter palavras que ache relevantes aparecerem na notícia. Estas vão depender do seu gosto e das suas preferências. Outro *HashSet* utilizado é o *textNot*, que por sua vez contem as palavras presentes no parágrafo da notícia.

Através do método *retainAll* é feita a interseção dos dois *HashSet* e a partir daqui é possível relacionar o texto da notícia com o léxico. Este método devolve apenas o conjunto de *strings* comuns nos dois conjuntos de dados. Sendo o *lexSize* o tamanho do *lexicoRelevenate*, usado para calcular a percentagem final de quantas palavras do léxico estão presentes na notícia. A seguinte expressão demonstra o cálculo efetuado no algoritmo:

$$\mu = \frac{textNot \cap lexicoRelevante}{lexSize} \tag{4.1}$$

Se ocorrer que,

$$\mu > 35\% \tag{4.2}$$

a notícia será destacada. Optou-se por 35%, pois apesar de existir um léxico para cada tema, dentro do mesmo tema podem existir várias palavras que não faz sentido aparecerem na mesma notícia.

De seguida são demonstrados alguns exemplos, atribuindo valor a cada um dos *HashSet*.

textNot	**lexicoRelevante**	**retainAll**
primeiro	economia	Costa
ministro	parlamento	António
António	petróleo	sucesso
Costa	Angola	cimeira
considera	António	ministro
cimeira	Costa	europeia
europeia	cimeira	primeiro
um	europeia	
sucesso	investimento	
	orçamento	

Tabela 4.1: Exemplo de interseção de dois léxicos.

Na tabela 4.1 observamos o resultado do método *retainAll*, dado dois léxicos, um com a notícia e outro com o *léxicoRelevante*. Podemos concluir que a notícia presente no exemplo será destacada.

Figura 4.1: Exemplo de notícia destacada.

Na figura 4.1 é possível ver a diferença que existe para o utilizador de uma notícia que foi destacada, para as restantes. Aqui obtemos o efeito pretendido, uma vez que a notícia desperta a atenção do utilizador, sendo fácil localizar primeiro as notícias destacadas.

4.2.3 getParagraph

Esta função é muito relevante no contexto do projeto, uma vez que permite simplificar a notícia que será mostrada. Para isto é usado um algoritmo bastante simples mas muito eficaz.

```
Document doc = Jsoup.connect(url).get();
Elements paragrafos = doc.select("p");
StringBuilder sb = new StringBuilder();
    for (Element paragrafo : paragrafos) {
        String t = paragrafo.text().trim();
        if (t.length() == 0) {
                continue;
        }
        char last = t.charAt(t.length() - 1);
        if (".!?".contains(last + "")) {
            sb.append(t);
            if (sb.length() > 200) {
                    break;
            }
        }
    }
```

Listing 4.3: Código usado para destacar notícia

Com recurso a *Jsoup* escolhe-se o primeiro parágrafo, *p*, do documento HTML. O próximo passo é escolher apenas aqueles cujo último caráter seja um dos seguintes sinais de pontuação: ., ! ou ?. Este algoritmo é simples mas extremamente eficaz, uma vez que todos os parágrafos de uma notícia terminam com alguns destes sinais de pontuação, neste caso será guardado o primeiro parágrafo por ser considerado o que contem informações mais relevantes dentro de uma notícia. Esta seleção de parágrafos mostrou-se importante, uma vez que algumas páginas *web* não usam as *tags p* para mostrar os parágrafos da notícia, houve casos em que o primeiro parágrafo do *website* dizia respeito ao autor do texto e não propriamente do texto da notícia.

4.3 Conclusões

Neste capítulo foram apresentadas algumas funções usadas, bem como o resultado da sua utilização.

Capítulo 5

Conclusões e Trabalho Futuro

5.1 Introdução

Neste capítulo são apresentadas as principais conclusões sobre o trabalho desenvolvido, bem como o trabalho futuro que pode ser desenvolvido.

5.2 Conclusões Principais

Pode-se concluir que existem métodos eficazes que permitem retirar a informação presente na *web*, estes seguem regras que se mostraram eficazes.

Através de experiências realizadas e implementadas, como apresentadas em *getparagraph* (4.2.3), provou ser um método eficaz, pois em grande parte dos *websites* utilizados, foram poucos os casos em que não devolveu um parágrafo válido para a notícia.

Este projeto provou também que existe uma forma diferente de ver notícias, pois podem ser observadas de uma forma resumida e onde se destacam conteúdos que o utilizador considera relevantes.

5.3 Trabalho Futuro

Grande parte do trabalho futuro passa pelo estudo e desenvolvimento de novas técnicas de aceder a *websites* com conteúdo noticioso. A aplicação usa o recurso ao *website*, www.sapo.pt, se a estrutura do cógido HTML for alterada, os métodos *Jsoup* necessitam de ser alterados, tais como, as *querys* ao documento HTML. Também passará por melhorar os métodos desenvolvidos, uma vez que alguns deles são muito pesados computacionalmente, idealmente era bom usar métodos de

forma a poupar recursos e aumentando a velocidade no processamento da informação.

Bibliografia

[1] Class Selector.

[2] JavaFx. https://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm.

[3] Jsoup. https://jsoup.org/.

[4] SceneBuilder. http://www.oracle.com/technetwork/java/javase/downloads/sb2download-2177776.html.

[5] Web Scraping. https://en.wikipedia.org/wiki/Web_scraping.

[6] Gonçalves D Fonseca M., Campos P. *Introdução ao Design de Interfaces*. FCA, 2012.

[7] Gao W. Chin S. Iverson D. Vos J Weaver, J. *Pro JavaFx 8*. appress, 2014.