



IGOR HORTA CORREA

UMA FERRAMENTA DE *E-COMMERCE*
PARA ELABORAÇÃO E COMPRA DE
COMPUTADORES COM *HARDWARE*
CUSTOMIZADO

LAVRAS - MG

2015

IGOR HORTA CORREA

**UMA FERRAMENTA DE *E-COMMERCE* PARA
ELABORAÇÃO E COMPRA DE COMPUTADORES COM
HARDWARE CUSTOMIZADO**

Monografia apresentada ao Colegiado do
Curso de Sistemas de Informação, para
a obtenção do título de Bacharel em
Sistemas de Informação.

Orientador

MSc. Renato Resende Ribeiro de Oliveira

Coorientador

Dr. Denilson Alves Pereira

LAVRAS - MG

2015

**Ficha Catalográfica Elaborada pela Coordenadoria de Produtos
e Serviços da Biblioteca Universitária da UFLA**

Correa, Igor Horta.

Uma ferramenta de *e-commerce* para elaboração e compra de computadores com *hardware* customizado/ Igor Horta Correa. – Lavras : UFLA, 2015.

71 p. : il.

Monografia (graduação) – Universidade Federal de Lavras, 2016.

Orientador: Renato R. R. de Oliveira.

Bibliografia.

1. Extração de dados da web. 2. E-Commerce. 3. Web. 4. Banco de dados. I. Universidade Federal de Lavras. I. Título.

CDD – 004.6

IGOR HORTA CORREA

**UMA FERRAMENTA DE *E-COMMERCE* PARA
ELABORAÇÃO E COMPRA DE COMPUTADORES COM
HARDWARE CUSTOMIZADO**

Monografia apresentada ao Colegiado do
Curso de Sistemas de Informação, para
a obtenção do título de Bacharel em
Sistemas de Informação.

APROVADA em 24 de janeiro de 2016.

Dr. Denilson A. Pereira UFLA

Dr. Ahmed A. A. Esmin UFLA

Dr. Raphael W. de Bettio UFLA

MSc. Renato Resende Ribeiro de Oliveira
Orientador

LAVRAS - MG

2015

O presente trabalho

AGRADECIMENTOS

Agradeço a.

Agradeço também a.

RESUMO

Resumo da monografia

Palavras-chave: E-Commerce. Tecnologia da Informação. Extração de Dados. Mineração de Dados.

ABSTRACT

The english translation.

Keywords: E-Commerce. Information Technology. Data Extraction. Data Mining.

LISTA DE FIGURAS

Figura 1	Exemplo de documento em HTML.	16
Figura 2	Interpretação de código HTML pelo browser Google Chrome.	16
Figura 3	Código em HTML com trecho em Javascript.	18
Figura 4	Exemplo de código em Javascript que pode ser executado pelo PhantomJS.	19
Figura 5	Imagem gerada após a execução do código Javascript pelo PhantomJS.	20
Figura 6	Exemplo de uma tabela de um banco de dados relacional.	22
Figura 7	Exemplo de uma tabela de lojas do banco de dados.	26
Figura 8	Exemplo de uma tabela de produtos de um banco de dados.	27

LISTA DE TABELAS

LISTA DE SIGLAS

HTML	Hypertext Markup Language
JS	JavaScript
JVM	Java Virtual Machine
SGBD	Sistema Gerenciador de Banco de Dados
MVC	Model View Controller
XML	Extensible Markup Language
DOM	Document Object Mode

SUMÁRIO

1	INTRODUÇÃO	12
1.1	Contextualização e Motivação	12
1.2	Objetivos Gerais e Específicos	13
1.3	Metodologia.....	14
1.4	Organização do Trabalho	14
2	REFERENCIAL TEÓRICO	15
2.1	Hypertext Markup Language (HTML)	15
2.2	Javascript	17
2.3	PhantomJS.....	19
2.4	Java	20
2.5	MySQL	21
2.6	Hibernate	23
2.7	VRaptor	23
3	METODOLOGIA	25
3.1	Extração de dados sobre produtos	25
3.2	Persistência dos dados	26
3.3	Listagem de produtos com comparação de preços.....	27
4	RESULTADOS E DISCUSSÃO	28
4.1	Cronograma.....	28
4.2	Resultados Esperados.....	28
5	CONCLUSÃO E TRABALHOS FUTUROS.....	29
	REFERÊNCIAS.....	30

1 INTRODUÇÃO

1.1 Contextualização e Motivação

Com o advento da Internet, surgiu também uma nova espécie de comércio, o chamado *e-commerce* ou comércio digital, que possui a vantagem de não limitar geograficamente o consumidor e disponibilizar para o mesmo um maior conjunto de lojas e produtos. De acordo com a E-bit, as vendas no comércio eletrônico em 2014 no Brasil continuaram crescendo e atingiram um resultado além do esperado. O faturamento do setor *e-commerce* arrecadou com vendas o equivalente a R\$35,8 bilhões. O que representa um crescimento de 24% em relação a 2013, quando se vendeu um total de R\$28,8 bilhões (EBIT, 2015).

O número de pedidos também aumentou, segundo a (EBIT, 2015), o aumento deste em 2014 foi de 17% em relação ao ano anterior, chegando a 103,4 milhões. Em 2013 foram 88,3 milhões de encomendas de bens de consumo via Internet. Em 2015, espera-se que o número de encomendas seja 19% maior, chegando a 122,9 milhões.

Com o comércio digital em ascensão e consequentemente o aumento das lojas virtuais e das diversas opções de compra, fica cada vez mais difícil e inviável para o usuário realizar a comparação de preços entre diversas lojas por si só. Tal tarefa é ainda mais difícil para quem deseja montar um computador personalizado do zero, o que requer que o mesmo pesquise e compare preços de pelo menos 10 peças em diversos sites de *e-commerce* especializados.

Tendo em vista esta dificuldade surge a necessidade de um sistema que realize a comparação de preços entre diversas lojas de peças de com-

putador, que auxilie também o usuário a verificar a compatibilidade das mesmas. Tal sistema utilizará técnicas de extração de dados para obter preços e informações sobre diversas peças de computadores de diferentes sites especializados em comércio eletrônico. Com tais informações disponíveis o sistema proposto irá disponibilizar diversos recursos como comparação de preços entre peças e auxílio na customização de computadores visando o melhor custo benefício.

1.2 Objetivos Gerais e Específicos

O objetivo geral deste trabalho é construir uma aplicação web que auxilie o consumidor a comprar e montar seu computador personalizado. Esta aplicação possuirá o recursos de comparação de preços de peças de computadores das principais lojas de *e-commerce* desta categoria.

Visando atingir este objetivo geral, os objetivos específicos são definidos a seguir:

- Desenvolver um algoritmo eficiente que seja capaz de coletar informações de produtos em diferentes sites de *e-commerce*.
- Extrair as informações sobre produtos de pelo menos 5 sites que vendem peças de computadores.
- Modelar e criar o banco de dados da aplicação, integrando as informações coletadas dos diferentes sites com os dados internos da aplicação desenvolvida.
- Desenvolver um protótipo de interface para aplicação (*website*).
- Implementar e tornar disponível para os consumidores um sistema capaz de montar um computador customizado e buscar os melhores preços para todas as peças.

1.3 Metodologia

Será desenvolvida uma aplicação para realizar comparações de preços entre peças de computadores de diversas lojas de *e-commerce*, tal aplicação aprestará recursos para auxiliar o usuário a montar seu próprio *desktop* customizado com o melhor custo benefício.

Um algoritmo para extração de dados será utilizado para se obter informações como: preço, imagem e descrição de produtos, tal algoritmo será escrito em Javascript e será executado utilizando-se o PhantomJS. O *script* obterá a árvore DOM do *website* e aí então buscará por padrões que possam identificar uma lista de produtos. Se tal lista for encontrada com sucesso a mesma será enviada para a aplicação, que será responsável por verificar a integridade dos dados e então salva-la no banco de dados.

Depois de obter os dados o próximo passo será criar uma interface amigável, tal interface será criada utilizando-se HTML, CSS e Javascript e possuirá recursos para listar os produtos obtidos com seus respectivos preços.

1.4 Organização do Trabalho

O trabalho está organizado da seguinte forma, na Seção 2 será apresentado um referencial teórico acerca dos conceitos utilizados. Na Seção 3 será apresentada a metodologia do trabalho desenvolvido. Na Seção 4 serão apresentadas as simulações computacionais realizadas e os resultados obtidos. As conclusões e os trabalhos futuros serão apresentadas na Seção 5.

2 REFERENCIAL TEÓRICO

2.1 Hypertext Markup Language (HTML)

O *Hypertext Markup Language* (HTML) é uma linguagem de marcação utilizada para descrever documentos da web (*websites*). Linguagem de marcação é um conjunto de *tags* (palavras reservadas da linguagem) e regras que ao serem aplicados em um arquivo de texto provêm informações sobre como o documento deve ser editado, formatado, exibido e impresso.

A *World Wide Web* sempre utilizou o HTML como linguagem de marcação. Inicialmente o HTML foi desenvolvido para suprir a necessidade de descrever semanticamente documentos acadêmicos. Entretanto ao passar dos anos este sofreu significativas adaptações e melhorias para suportar novos tipos de documentos.

Em HTML, as *tags* normalmente são utilizadas em duplas, onde cada uma indica o início e fim do texto que está sendo demarcado, como por exemplo:

```
<title>Olá mundo</title>
```

Ao interpretar este trecho de HTML, o navegador do usuário que está acessando o documento irá entender que o título da página deve ser “Olá mundo”.

A Figura 1 mostra um exemplo de um documento escrito em HTML. Segundo W3C (2012) o HTML é dividido em 2 partes principais, *head* (cabeçalho) e *body* (corpo). O título do documento juntamente com demais informações sobre o mesmo ficam dentro da *tag* `<head>`. Dentro da *tag* `<body>` se encontra o conteúdo (corpo) do documento. Já as *tags* `<h1>`, `<h2>`, `<h3>` representam tópicos no documento, onde `<h1>` é um elemento

mais importante que `<h2>` e que `<h3>`. Os navegadores normalmente exibem os tópicos mais importantes com fontes maiores do que tópicos menos importantes. A Figura 2 apresenta a interpretação deste exemplo pelo navegador *Google Chrome*.

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <title>
6       Titulo da Pagina
7     </title>
8   </head>
9   <body>
10    <div class='container'>
11      <h1>Titulo grande</h1>
12      <h2>Titulo medio</h2>
13      <h3>Titutlo pequeno</h3>
14    </div>
15  </body>
16 </html >

```

Figura 1 Exemplo de documento em HTML.

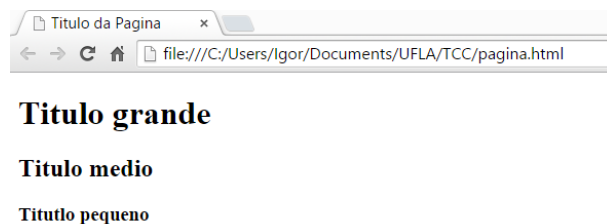


Figura 2 Interpretação de código HTML pelo browser Google Chrome.

Como o HTML é a linguagem utilizada para descrever documentos da *web*, o algoritmo proposto neste trabalho precisará ser capaz de ler, interpretar e processar códigos em HTML para recuperar as informações necessárias.

A aplicação proposta por este estudo para realizar a comparação de preços de diferentes lojas de peças de computadores também apresentará informações utilizando o HTML.

2.2 Javascript

O *Javascript* é uma linguagem de programação *script* que é muito utilizada atualmente para criação de páginas da web dinâmicas. Também é muito utilizada para construção de aplicações web. Isso se deve ao fato de os navegadores modernos suportam por padrão o uso dessa linguagem pelos *websites*.

Segundo (ECMA International, 2011) linguagem de *script* é uma linguagem de programação usada para manipular, customizar e otimizar recursos já existentes em um programa. Em tais sistemas, funcionalidades úteis já são disponibilizadas através de uma interface para o usuário. A linguagem de *script* é um mecanismo para disponibilizar tais funcionalidades também para o controle da aplicação.

De acordo com (FLANAGAN, 2002), Javascript é uma linguagem de *script* com suporte a orientação a objetos. Na programação orientada a objetos implementa-se um conjunto de classes que definem os objetos presentes na aplicação. Cada classe determina o comportamento (definido nos métodos) e estados possíveis (atributos) de seus objetos, assim como o relacionamento com outros objetos.

Quando utilizada no *client-side* (lado do cliente), o Javascript permite que conteúdo executável seja adicionado às páginas *web*, o que possibilita que estas páginas interajam com o usuário, controlem o navegador e criem conteúdo HTML dinamicamente.

```

1 <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
2 "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
3 <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
4   <head>
5     <title>
6       Titulo da Pagina
7     </title>
8   </head>
9   <body>
10    <script type="text/javascript">
11      document.write("Hello World from JavaScript!");
12    </script>
13  </body>
14 </html >

```

Figura 3 Código em HTML com trecho em Javascript.

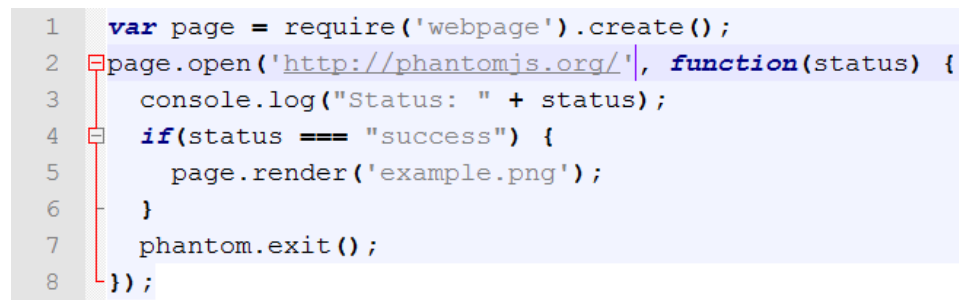
A Figura 3 apresenta um documento em HTML que possui um trecho de código em Javascript. Este trecho se encontra dentro da tag `<script>`, isso indica ao navegador que este trecho trata-se de um código em Javascript. Na linha 11 pode-se observar a chamada da função *write* do objeto *document*. Esta função tem como objetivo adicionar dinamicamente um texto ou conteúdo qualquer à página que está sendo exibida pelo navegador.

Devido aos diversos recursos da linguagem Javascript para promover dinamicidade a *websites*, esta se tornou amplamente utilizada em diversos *websites* modernos, inclusive em sites voltados para o *e-commerce*. Como foi discutido no trabalho de (ALONSO, 2012), a atualização de preços de produtos dinamicamente via Javascript em lojas virtuais pode ser um problema para algoritmos de extração de dados. No presente trabalho este problema sera tratado e formas de soluçona-lo serão apresentadas.

2.3 PhantomJS

O PhantomJS é um navegador de código aberto sem interface gráfica (*headless*) que utiliza o motor *webkit*. Códigos podem ser desenvolvidos em Javascript para que o PhantomJS os execute. Ele disponibiliza à estes códigos suporte para diversos padrões web, como manipulação da DOM, seletores CSS, JSON, Canvas e SVG. Utilizando o PhantomJS é possível:

- Realizar testes unitários para aplicações web.
- Carregar, analisar e renderizar páginas web.
- Obter *screenshots* de *websites*.

A imagem mostra um trecho de código JavaScript em um editor de texto com fundo azul claro. O código está numerado de 1 a 8 à esquerda. O código cria uma instância da classe 'webpage', abre o URL 'http://phantomjs.org/' e, se o status for 'success', renderiza um arquivo 'example.png'. O código termina com 'phantom.exit()' e '});'.

```
1 var page = require('webpage').create();
2 page.open('http://phantomjs.org/', function(status) {
3     console.log("Status: " + status);
4     if(status === "success") {
5         page.render('example.png');
6     }
7     phantom.exit();
8 });
```

Figura 4 Exemplo de código em Javascript que pode ser executado pelo PhantomJS.

O código apresentado na Figura 4 associa uma instância da classe *webpage* do PhantomJS à uma variável chamada *page*. Em seguida a função *open* é chamada. Esta função realiza uma requisição ao site fornecido como parâmetro, no caso “http://phantomjs.org/”. Se a requisição for um sucesso, uma cópia da imagem renderizada pelo navegador PhantomJS é criada. A Figura 5 é a imagem gerada pela execução deste exemplo.

O PhantomJS foi a ferramenta escolhida para executar o algoritmo de extração de dados que este estudo propõe, entre os diversos motivos desta

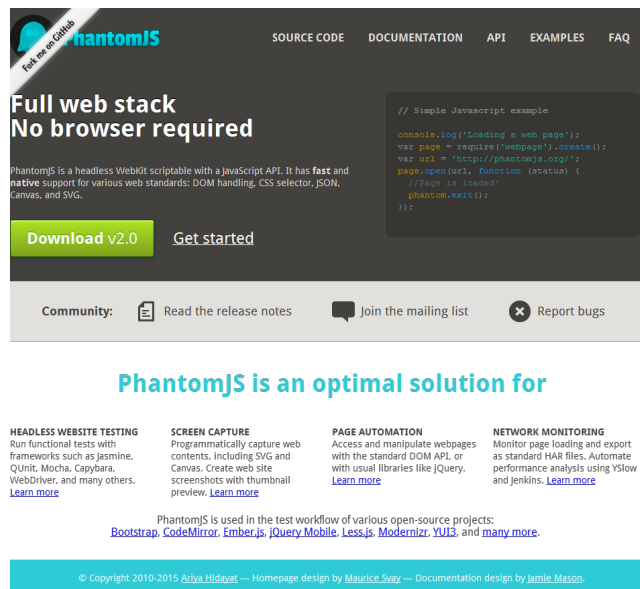


Figura 5 Imagem gerada após a execução do código Javascript pelo PhantomJS.

escolha se destaca o fato da ferramenta oferecer total suporte a linguagem Javascript, linguagem na qual o algoritmo proposto será desenvolvido, outro motivo é o fato do Phatom.JS possuir diversas bibliotecas extras que navegadores comuns não possuem.

2.4 Java

A linguagem de programação Java começou a ser desenvolvida em 1991 com a finalidade de ser utilizada em dispositivos eletrônicos inteligentes. Entretanto com o crescimento exponencial da *World Wide Web* em 1993, os engenheiros da *Sun Microsystems* notaram o potencial da linguagem para aprimorar a funcionalidade de servidores *web* e a adaptou para este propósito. DEITEL e DEITEL (2006)

A linguagem Java é orientada a objetos e possui o diferencial de

ser interpretada. Diferentemente de linguagens como C e C++, onde os códigos-fonte são diretamente transformados em códigos de máquina quando compilados, o compilador Java transforma o código-fonte em uma linguagem intermediária. Esta é então interpretada em tempo de execução por uma máquina virtual (programa denominado *Java Virtual Machine*). Como resultado, programas escritos em Java podem muitas vezes atingir um nível de eficiência que é inatingível com outras linguagens puramente interpretadas tradicionais. Roberts (2013)

Segundo Roberts (2013) as características principais do Java incluem o fato dela ser fácil de ser utilizada, sem a necessidade de treinamento extensivo do desenvolvedor. Em Java estes desenvolvedores têm acesso à diversas bibliotecas que oferecem uma série de recursos já implementados, tais como operações de entrada e saída, interfaces para redes e recursos gráficos. Por ser uma linguagem interpretada, uma outra vantagem é o fato dela possuir uma verificação de erros em tempo de execução muito mais robusta do que linguagens não interpretadas (como C e C++).

Devido as características citadas acima, nossa aplicação terá o *backend* (parte do sistema que é executado por um servidor) implementada em Java

2.5 MySQL

SGBD ou (sistema gerenciador de banco de dados), é um conjunto de programas que tem como objetivo armazenar, alterar, apagar e proteger informações de uma base de dados. Estes são bastante úteis em aplicações web, por permitirem que as informações sejam armazenadas e utilizadas automaticamente.

Banco de dados armazenam informações sobre objetos distintos, entidades, associações ou até mesmo sobre o relacionamento entre estas entidades. Como por exemplo, um banco para vendas pode guardar informações sobre produtos, clientes e vendas. Neste caso produtos e clientes são entidades enquanto vendas é um relacionamento entre cliente e produto.

No modelo de dados relacional os dados são percebidos pelos usuários como tabelas, onde cada coluna armazena um tipo de dados que podem ser de diversos tipos diferentes como numéricos, reais, texto e etc. Cada linha desta tabela representa uma instância.

Na figura 6: podemos ver como os dados são representados em um modelo relacional, a coluna `ClienteID` representa o identificador do cliente este número deve ser único, a segunda coluna representa o nome e a terceira o telefone celular de cada uma das linhas correspondentes.

<u>ClienteID</u>	Nome	Celular
1	Regina Ribeiro	35-97412145
2	Felipe Alvarenga	35-91421002

Figura 6 Exemplo de uma tabela de uma banco de dados relacional.

O MySQL segundo (TAHAGHOGHI; WILLIAMS, 2006) é um popular SGBD relacional *openSource*, que se destaca dos demais concorrentes por uma série de motivos, dentre elas podemos ressaltar: seu tamanho e velocidade, o MySQL funciona bem até mesmo nos hardwares mais modestos, a velocidade com que ele consegue recuperar informações foi fator fundamental para torná-lo o SGBD favorito em diversas organizações. Outro fator muito importante para sua aceitação é a sua fácil instalação que pode ser realizada sem grandes dificuldades e com pouca configuração

2.6 Hibernate

Segundo (IRELAND et al., 2009), enquanto o paradigma relacional é um modelo extremamente comum em populares banco de dados do mercado, a programação orientada a objetos também é extremamente comum em aplicações atuais, logo a combinação entre estas tecnologias de diferentes paradigmas se torna inevitável. Enquanto o modelo relacional trata os dados como tabelas, o modelo orientado a objetos os representa como um conjunto de objetos, tamanha diferença em como os dados são representados geram uma série de problemas em aplicações que utilizam SGBDS relacionais e linguagens orientadas a objeto.

Como uma alternativa para este problema, algumas ferramentas de mapeamento objeto/relacional(MOR) foram criadas, dentre essas destaca-se o Hibernate

O Hibernate é um *framework*(ferramenta) de alta performance para mapeamento objeto/relacional em java. Este *framework* associa classes em java a tabelas do banco de dados. Ele também permite abstrair código sql em objetos java. Entre as diversas vantagens do Hibernate podemos ressaltar a facilidade em alterar formatos das tabelas, simplesmente alterando as classes java que as representam.

2.7 VRaptor

Model View Controller(Modelo Visão controlador) é um padrão de arquitetura de *software*, que separa a aplicação em três partes. O modelo (*model*) representa os dados da aplicação e suas respectivas regras de negócios. A visão (*view*) pode ser qualquer saída de representação dos dados.O controlador(*controller*) faz a mediação da entrada, preparando-a em coman-

dos para o modelo ou visão. O objetivo principal do MVC é trazer para aplicação reusabilidade de código e separação de conceitos.

O Vraptor é um *framework MVC* de código livre focado em gerar alta produtividade, foi designado para dar suporte ao desenvolvimento de *web-sites* dinâmicos bem como aplicações *web* que fazem uso do modelo *MVC*. Tem como objetivo poupar trabalho do desenvolvedor, mantendo-o focado nas suas atividades principais. De acordo com (CAVALCANTI, 2014), com o Vraptor é fácil aplicar as melhores práticas de orientação a objetos, já que este não impõe restrições no design das suas classes. É também um dos frameworks mais extensíveis em java, pois praticamente todos seus comportamentos podem ser sobrescritos com pouco código, sem a necessidade de configurações em *XML*. Além disso, é um *framework* que deixa seus usuários totalmente livres para escolher sua camada de visualização. Dentre as diversas vantagens citadas acima, podemos destacar o fato do Vraptor ser um *framework* nacional com ampla documentação em português.

3 METODOLOGIA

3.1 Extração de dados sobre produtos

(ALONSO, 2012) realizou um estudo sobre extração de preços em sites de *e-commerce*. Neste trabalho foi desenvolvido um agente de *Shopping Comparison* que utilizava padrões em documentos HTML para obter produtos e seus respectivos atributos. Porém este estudo possuía o ponto deficiente de não conseguir obter informações de *websites* que utilizam a linguagem Javascript para exibir o preços dos produtos assincronamente, o que é uma prática comum em sites atuais.

O presente trabalho tratará os pontos deficientes do estudo de Alonso (2012), e também abordará outras funcionalidades da aplicação em si.

Um dos recursos do sistema proposto é proporcionar a comparação de preços de peças de computador entre diversas lojas de *e-commerce*, para se obter a informação de cada peça com seus respectivos preços e vendedores o sistema utilizara de um algoritmo de extração de dados. Tal algoritmo será desenvolvido em Javascript e será executado utilizando o PhantomJS. Os dados que deveram ser extraídos dos *websites* voltados para o *e-commerce* são:

- nome do produto.
- link da imagem do produto.
- preço do produto.
- descrição do produto.
- link para a compra do produto.

As informações do produto como nome, *link* da imagem, preço, *link*

para a compra serão necessárias para identificar o produto, exibir sua imagem e manter informações sobre o vendedor.

Para se obter tais informações será criado um *script* que adotara a seguinte estratégia: obter a árvore DOM do *website* utilizando funções disponibilizadas pelo phantomJS e ai então utilizando a DOM juntamente com seletores CSS o algoritmo ira buscar pelo carácter \$ pois o mesmo denota preço, ao encontra-lo analisara os elementos próximos do mesmo nível em busca de uma imagem e nome do produto. Se tais elementos forem localizados, o algoritmo então conseguiu identificar um possível produto. Para confirmar que o item obtido é o que se procura, o algoritmo ira subir alguns níveis na arvore DOM e então procurar nos elementos próximos por itens semelhantes, se uma sequência de produtos semelhantes forem encontrados, o algoritmo então ira armazenar estes itens em uma lista e os enviara para a aplicação.

3.2 Persistência dos dados

O sistema proposto possuirá 2 tabelas essenciais no banco de dados, sendo a primeira de lojas, que conterà informações sobre as lojas de *e-commerce*, a segunda tabela contará com informações sobre os produtos e será preenchida com os dados obtidos por meio do algoritmo de extração de dados, tais dados serão tratados e terão sua integridade testada antes de serem salvos.

Id	Nome	Link
1	Kabum	http://www.kabum.com.br/hardware/

Figura 7 Exemplo de uma tabela de lojas do banco de dados.

Id	Nome	Imagem	status	Categoria	Id_loja	Descrição
1	Processador i7	http://kabum.com.br/produtos/fotos/i7.jpg	Em estoque	Processador	1	Processador i7 4510 2.0GHz

Figura 8 Exemplo de uma tabela de produtos de um banco de dados.

A figura 7 representa uma tabela de lojas no banco de dados, tal tabela conterá o nome, *link* para o site da loja e id de cada empresa que terá seus dados extraídos.

A figura 8 representa uma tabela de produtos no banco de dados, esta tabela ira conter informações como: id,nome,imagem,status,categoria,descrição e um identificador que informara a qual loja este produto pertence.

3.3 Listagem de produtos com comparação de preços

Com os dados extraídos de diferentes lojas de *e-commerce* voltadas para o comércio de hardware, sera possível a implementação de uma interface *web* que auxilie ao usuário na montagem do próprio computador personalizado. tal interface sera implementada utilizando-se as linguagens HTML, Javascript e CSS.

Nela o usuário terá acesso a diferentes tipos de peças de computadores, o sistema destacará a peça mais barata de cada categoria e seu respectivo vendedor.

4 RESULTADOS E DISCUSSÃO

4.1 Cronograma

- 1 - Estudar sobre algoritmos de detecção de padrões em páginas HTML (1 mês)
- 2 - Desenvolver um algoritmo eficiente para reconhecimento de padrões em HTML, que trate alterações na página com Javascript (2 meses)
- 3 - Desenhar e planejar as telas da aplicação (15 dias)
- 3 - Modelagem do banco de dados para a aplicação (15 dias)
- 4 - Implementação do *back-end* da aplicação (1 mês)
- 5 - Implementação do *front-end* da aplicação (1 mês)

4.2 Resultados Esperados

Ao final deste trabalho espera-se um algoritmo que seja capaz de extrair preços e demais informações de pelo menos 5 sites de *e-commerce* diferentes, sendo que pelo menos 2 deles utilizem a linguagem Javascript para atualizar as informações dos produtos assincronamente.

Também espera-se um protótipo de uma aplicação *web* que utilize os dados obtidos pelo algoritmo de extração de dados. Tal aplicação deve ter como objetivo comparar os preços dos produtos obtidos pelo algoritmo.

5 CONCLUSÃO E TRABALHOS FUTUROS

REFERÊNCIAS

ALONSO, L. S. *Agente inteligente de shopping comparison através de detecção de padrões*. 2012.

CAVALCANTI, L. *VRaptor, Desenvolvimento ágil para Web com JAVA*. 2^a. ed. São Paulo: Editora Casa do Código, 2014.

DEITEL, H.; DEITEL, P. *Java como programar. [Sl]*. [S.l.]: Editora Bookman, 2006.

EBIT. *Relatório WebShoppers*. 2015. Web site. Acessado em 10/06/2015. Disponível em: <<http://www.ebit.com.br/webshoppers>>.

ECMA International. *Standard ECMA-262 - ECMAScript Language Specification*. 5.1. ed. [s.n.], 2011. Disponível em: <<http://www.ecma-international.org/publications/standards/Ecma-262.htm>>.

FLANAGAN, D. *JavaScript: the definitive guide*. [S.l.]: "O'Reilly Media, Inc.", 2002.

IRELAND, C.; BOWERS, D.; NEWTON, M.; WAUGH, K. A classification of object-relational impedance mismatch. In: *Advances in Databases, Knowledge, and Data Applications, 2009. DBKDA '09. First International Conference on*. [S.l.: s.n.], 2009. p. 36–43.

ROBERTS, E. *Art and Science of Java*. [S.l.]: Pearson Education Limited, 2013.

TAHAGHOGHI, S. M.; WILLIAMS, H. E. *Learning MySQL*. [S.l.]: "O'Reilly Media, Inc.", 2006.

W3C. *HTML5 : a vocabulary and associated APIs for HTML and XHTML*. 2012. Acesso em 20/05/2015. Disponível em: <<http://www.w3.org/TR/html5/>>.