Um *Web Service* para busca de Preços de Produtos

Claudinei Rodrigues Junior[1], Simone Nasser Matos[1]

[1]Coordenação de Informática – Universidade Tecnológica Federal do Paraná (UTFPR)
84016-210 – Ponta Grossa – PR – Brasil

Resumo. *Atualmente não há no mercado um software que implemente diversos métodos de formação de preço de venda e que seja gratuito. Conscientes deste problema, o Grupo de Pesquisa em Engenharia de Software (GPES) da Universidade Tecnológica Federal do Paraná (UTFPR), Campus Ponta Grossa, está desenvolvendo um framework que implementa diversos métodos de formação de preço de venda. Um dos métodos que será implementado, denominado método baseado nas decisões das empresas concorrentes, depende dos preços de venda praticados pelo mercado. Este trabalho visa desenvolver um Web Service que busque os preços de vendas de produtos em sítios de e-commerce para auxiliar o GPES na implementação deste método. Além disso, o Web Service permite o uso automatizado por aplicações e é de código aberto.*

1. Introdução

A competitividade do mercado é intensa e definir preços competitivos que agradem aos consumidores e permitam às empresas obterem lucros é uma tarefa complexa. A sobrevivência de uma empresa está diretamente ligada aos preços praticados. Um preço de venda mal-definido pode comprometer desde seu crescimento até sua estabilidade sócio-econômica.

Ao longo dos anos, diversos estudiosos de administração, tais como Santos (2001) e Sardinha (2005), criaram métodos para o cálculo do preço de venda. Cada método utiliza um conjunto de variáveis distintas e é adequado para situações específicas.

Cabe aos gestores conhecer os métodos, aplicá-los, analisar os resultados e decidir pelo melhor preço. Entretanto, o mercado é agressivo. Há diversos outros fatores que um gestor deve considerar além do preço e nem sempre sobra tempo para realizar os cálculos.

Atualmente não há alternativa aos gestores senão realizar os cálculos manualmente, pois os *softwares* disponíveis no mercado implementam somente alguns métodos e, em sua maioria, são pagos.

Conscientes do problema enfrentado pelos gestores e da falta de *softwares* que os auxiliem a tomar a decisão pelo preço de venda adequado mais agilmente, o Grupo de Pesquisa em Engenharia de *Software* (GPES) [GPES, 2010], da Universidade Tecnológica Federal do Paraná (UTFPR), Campus Ponta Grossa, vem desenvolvendo um *framework*, denominado Framemk, que implementa alguns métodos de formação de preço de venda. Este *framework* será gratuito e poderá ser utilizado por qualquer um que precise definir preços de vendas.

Este artigo apresenta o *web service* desenvolvido para colaborar com o projeto do GPES na implementação do método conhecido por "método baseado nas decisões das empresas concorrentes" [SARDINHA, 1995]. Este método utiliza como parâmetros de entrada os preços de um produto praticados pelo mercado e o *web service* desenvolvido neste trabalho é uma solução para o GPES no que tange à obtenção dos preços de um dado produto, pois, apesar de existirem sítios que relacionam os preços de produtos na internet, tais como BuscaPe [BUSCAPE, 2010], JaCotei [JACOTEI, 2010], entre outros, estes não disponibilizam nenhum mecanismo ou meio de acesso para realizar as buscas de maneira automatizada e não são de códigos abertos para serem alterados por outros desenvolvedores. E não é legal, do ponto de vista jurídico, conforme foi constatado através de contato por e-mail com as empresas responsáveis por estes sítios, desenvolver uma aplicação que obtenha e utilize os preços relacionados por estes sítios que relacionam preços de produto na internet.

O presente artigo está dividido em cinco seções. A seção 2 discute a importância de uma correta formação de preço de venda e apresenta alguns aspectos que devem ser considerados na hora de definir o preço. A seção 3 discorre sobre o modelo de *web services* e as principais tecnologias relacionadas com o modelo. A seção 4 apresenta o *web service* desenvolvido para suprir as necessidades do GPES. E, por fim, a seção 5 relata as conclusões sobre o trabalho desenvolvido e faz algumas sugestões para trabalhos futuros.

2. Formação de Preço de Venda

A formação do preço de venda é fundamental para a sobrevivência de uma empresa no mercado. Há vários fatores a serem considerados para decidir qual o preço de venda ideal de um produto ou serviço. Entre eles pode-se destacar o posicionamento estratégico, demanda, mercado, fixação de preços para penetração, para concorrência ou maximização de retorno [OLIVEIRA e CREMA, 2009].

A determinação do preço de venda exige do gestor conhecimento sobre os fatores que dão origem ao preço de venda bem como seus custos. Algumas empresas não conseguem determinar de forma precisa os custos e despesas, tornando difícil a verificação posterior do lucro obtido com as vendas. Isto pode comprometer desde o crescimento da empresa até sua estabilidade econômico-financeira.

Segundo Oliveira e Crema (2009) "decisões referentes à definição do preço envolvem aspectos muitas vezes analisados de forma empírica, baseadas em informações subjetivas, sem qualquer embasamento teórico". Entretanto, a competitividade do mercado não permite que uma empresa forme seus preços de venda baseadas em informações subjetivas.

Há diversos aspectos que uma empresa deve considerar para formar o preço de venda de seu produto, como análise do ciclo de vida do produto, comportamento do consumidor, mensuração e previsão da demanda. Entre as maneiras destaca-se a análise de mercado. Realizar uma análise de mercado é elencar informações para conhecer profundamente o ambiente no qual a empresa está inserida [SANTOS, 2001].

Uma das etapas da análise de mercado é a análise da concorrência, que avalia a relação dos produtos com a organização [SARDINHA, 1995]. Informações sobre produto ou serviço, praça, gerenciamento, promoções, finanças e preços praticados são elencadas. O *web service* desenvolvido e apresentado neste artigo auxilia a análise da

concorrência, obtendo os preços de mercados praticados pela concorrência, pois, até o presente momento, não há no mercado um *software* que implemente diversos métodos de formação do preço de venda, obrigando os gestores a usarem vários *softwares* simultaneamente ou a realizarem os cálculos manualmente.

Devido a este problema, o GPES está desenvolvendo um *framework* que implementa vários métodos de formação do preço de venda, tais como: formação de preço de venda com base no custo pleno [NASCIMENTO e VARTANIAN, 1999], formação de preço com custeio baseado em atividades [NAKAGAWA, 1995], método do SEBRAE-PR [SEBRAEPR, 2010], entre outros. Um destes métodos, chamado de método baseado nas decisões das empresas concorrentes, analisa o ambiente externo a empresa e utiliza, entre outros parâmetros de entrada, os preços de venda praticados pelo mercado [SARDINHA, 1995]. Por isso, surgiu a necessidade de se construir um *web service* para busca de preço de venda de um determinado produto, afim de que GPES pudesse utilizá-lo para obter os preços de venda de um dado produto e calcular seu preço de venda utilizando o método baseado nas decisões das empresas concorrentes.

3. *Web Services*

Um *Web Service* é qualquer serviço disponível na Internet que use um sistema de mensagens com XML (*eXtensible Markup Language*) padronizado e não esteja preso à linguagem de programação ou sistema operacional. Este tipo de serviço permite que duas aplicações se comuniquem através de uma *interface* bem definida [CERAMI, 2002].

É desejável, ainda que não indispensável, que os *Web Services* possam ser facilmente encontrados através de mecanismos simples de busca, sejam auto-descritivos e usem padrões comuns da Internet. Se estes aspectos forem atendidos, a integração automática de aplicações se torna possível.

Para que os serviços possam ser facilmente encontrados, há um diretório comum onde são armazenadas informações sobre os serviços disponíveis. Este diretório é chamado de UDDI (*Universal Description, Discovery and Integration*) [CERAMI, 2002].

As descrições de serviços disponíveis no UDDI são documentos no padrão WSDL (*Web Service Definition Language*). Independente de a descrição estar disponível no UDDI, este possui, além de outras informações, a URL que aponta para o local onde está armazenada a descrição completa do serviço.

Nas seções seguintes detalham-se as tecnologias empregadas, conhecidas como WUST (WSDL, UDDI e SOAP – *Single Object Access Protocol – Technologies*). Estas tecnologias unidas a XML compõem a estrutura básica dos *Web Services*.

3.1. XML

A XML é o padrão adotado para o transporte de dados [ABINADER e LINS, 2006; CERAMI, 2002]. Simples de ser compreendida, fácil de ser transportada e compatível com diversas plataformas, a XML é a responsável por grande parte da interoperabilidade entre aplicações de ambientes heterogêneos permitida pelos *Web Services*.

A XML serve de base para as outras três tecnologias fundamentais dos *Web Services*: SOAP, UDDI e WSDL.

3.2. WSDL

A WSDL é a linguagem utilizada para criar uma auto-descrição pública do serviço [ABINADER e LINS, 2006; CERAMI, 2002]. Entre as informações que podem constar neste documento estão: o padrão utilizado para troca de mensagem (SOAP, JMS – *Java Message Service*, entre outros), protocolo de transporte (HTTP – *Hyper Text Transfer Protocol*, HTTPR – *Hyper Text Transfer Protocol Reliable*, DIME – *Direct Internet Message Encapsulation*, entre outros) e o endereço lógico (URL) do serviço.

A WSDL é baseada na XML e o formato das descrições é definido e validado utilizando XML *Schema* (XSD).

A possibilidade de criar descrições dos serviços e, com isso, facilitar a integração automatizada de aplicações, consiste em uma das principais vantagens dos *Web Services* sobre os outros modelos distribuídos.

3.3. UDDI

O UDDI é o diretório onde as informações sobre os *Web Services* podem ser armazenadas [ABINADER e LINS, 2006; CERAMI, 2002]. O diretório contém mecanismos que permitem a descoberta (procura) de serviços.

Uma das principais vantagens da utilização do UDDI é a garantia de que todos os *Web Services* disponíveis que atendam aos requisitos exigidos pelo solicitante tenham sido considerados durante o processo de escolha.

3.4. SOAP

O SOAP é o protocolo adotado pela maioria dos *Web Services* para troca de mensagens [ABINADER e LINS, 2006; CERAMI, 2002]. Baseado em XML, o SOAP é projetado para que possa ser utilizado sobre qualquer protocolo, incluindo SMTP (*Simple Mail Transfer Protocol*), HTTP e FTP (*File Transfer Protocol*).

O funcionamento do SOAP é similar ao do RPC (*Remote Procedure Call*) em que: a mensagem enviada pelo cliente deve informar qual método deverá ser invocado e os valores de seus parâmetros, se estes existirem. O serviço devolverá uma mensagem contendo o valor retornado pelo método. Esta abordagem deve ser cuidadosamente projetada para causar o menor acoplamento possível entre as aplicações, pois o servidor irá esperar parâmetros de tipos de dados específicos e um formato de mensagem rigorosamente definido. Se a linguagem ou plataforma do solicitante não possuir o tipo de dado esperado ou for incapaz de gerar a mensagem no formato requerido, a comunicação estará comprometida.

Uma alternativa ao modelo de comunicação baseado em RPC é o modelo fundamentado em troca de mensagens. Neste modelo, o servidor espera um documento inteiro para processar e não uma mensagem contendo uma sequência rigorosa de parâmetros. Se o modelo for implementado de maneira síncrona, o solicitante irá esperar algum retorno do serviço, seja somente uma confirmação do recebimento do documento ou algum valor ou documento. Porém, se a implementação for feita de maneira

assíncrona, o cliente não aguardará uma resposta, cabendo ao serviço decidir se algo deverá ser enviado ao cliente.

Em ambos os modelos de comunicação, o conteúdo da mensagem estará dentro de um envelope SOAP.

4. Um *Web Service* para busca de Preços de Produtos

Esta seção apresenta o *Web Service* desenvolvido para solucionar o problema do GPES. A seção 4.1 apresenta as funcionalidades do *Web Service*. A seção 4.2 explica como é feito o *parsing* dos HTMLs. A seção 4.3 descreve a arquitetura do serviço e a seção 4.4 apresenta o funcionamento do serviço utilizando como cliente uma aplicação *desktop*.

4.1. Características

O *Web Service* desenvolvido pode ser configurado através de uma *interface* administrativa para *web* desenvolvida com JSF (*Java Server Faces*) [JSF, 2010]. Através desta *interface* o usuário pode:

- Cadastrar novos sítios de *e-commerce*;

- Atualizar as informações dos sítios já cadastrados;

- Definir as configurações de *proxy*;

- Informar o número máximo de *threads* de busca de preço sendo executadas simultaneamente. Esta configuração é importante para otimizar o desempenho da aplicação em função do *hardware* e da largura de banda disponível.

O *Web Service* foi desenvolvido usando a API WSIT [WSIT, 2010]. A API JAX-WS [JAX-WS, 2010] é o núcleo da WSIT e fornece os métodos básicos para a criação e disponibilização de *Web Services* enquanto as extensões providas pela WSIT oferecem mecanismos para garantir interoperabilidade, otimização e confiabilidade de mensagens e suporte a transações.

A interface do *Web Service* disponibiliza o método *getProductPrice* que aceita como parâmetro o nome do produto do tipo *String*. Quando um consumidor invoca o método, o serviço recupera do banco de dados os sítios cadastrados bem como seus parâmetros de requisição. Estes parâmetros são os valores esperados pelo sítio e, normalmente, são compostos por um parâmetro para o nome do produto e outro para categoria do produto. Como a interface de serviço não permite especificar qual a categoria do produto, a aplicação atribui ao parâmetro "categoria" da requisição o valor que indique "todas as categorias" e, para saber qual o valor deste parâmetro, a aplicação faz uma busca no HTML procurando por palavras-chaves que indiquem todas as categorias, tais como "Todas as categorias", "Todas", "*All Categories*", entre outras. São, então, criadas as requisições e enviadas para os sítios.

O processo de criação das requisições é feita utilizando as classes nativas do Java do pacote *java.net*. Uma das informações extraídas do HTML quando um sítio cadastro é se o servidor espera requisições do tipo GET ou POST. Se for o primeiro caso, é feita uma requisição com os parâmetros sendo passados pela URL. Porém, se o servidor esperar requisições do tipo POST, um datagrama é criado contendo os valores dos parâmetros esperados pelo servidor. Sítios que utilizam buscas baseadas em Javascript e/ou JSON não são suportados pela aplicação.

Assim que são obtidos os HTMLs resultantes das requisições, eles são analisados pelo *parser*. Os preços e as descrições obtidas dos HTML são retornados pelo *Web Service* para o consumidor.

Todo o processo de criação das requisições, envio e análises dos retornos são feitas em paralelo até o número máximo de *threads* de requisição definido pelo usuário. Caso o usuário não tenha definido nenhum valor explicitamente, o valor padrão é dez.

4.2 Parsing dos HTMLs dos sítios de *e-commerce*

Sempre que um sítio é cadastrado, a aplicação obtém seu HTML (*HyperText Markup Language*) e realiza o *parsing* do documento. Este *parsing*, ou seja, a análise do documento e a criação de uma estrutura de dados com os elementos do documento, procura um formulário de busca baseado em palavras-chaves de vários idiomas, tais como "*search", "*procura", "busca-rápida*", "recherche"*, "*cerca",* entre outras. O fato de palavras-chaves de vários idiomas serem utilizadas na busca visa permitir que a aplicação possa realizar buscas não somente em sítios nacionais, mas também internacionais. Se um formulário de busca for encontrado, extraem-se (do HTML) os parâmetros necessários para realizar uma requisição. Caso contrário, o usuário é informado que o *web service* não pode realizar buscas nesse sítio e que ele não será cadastrado.

Quando é feita uma requisição para um sítio, a aplicação obtém o HTML resultante da requisição e realiza um *parsing* no documento para procurar as relações "descrição do produto" e "preço". Esta busca na estrutura de dados resultante do *parsing* é feita em duas etapas. Na primeira, procura-se pela primeira incidência de um *match* entre o nome do produto procurado pelo consumidor e um elemento do HTML. Quando ocorre um *match*, a aplicação armazena em qual posição a incidência ocorreu e busca o próximo *match* partindo da posição da primeira incidência. Assim que ocorre a segunda incidência, a aplicação inicia a próxima etapa do processo de busca, que consiste em procurar pelo preço do produto localizado no intervalo definido pelas posições do primeiro e segundo *match*. Para que o preço possa identificado pela aplicação, ele precisa ter um símbolo monetário, tal como US$ ou R$. Se for encontrado um preço, este será associado à descrição do produto do primeiro *match*. Entretanto, caso não seja encontrado um preço, a primeira incidência da descrição do produto será descartada. Este processo é repetido até que todo o HTML tenha sido analisado.

4.3. Arquitetura

A arquitetura do *Web Service* é composta por cinco pacotes básicos: *control*, *model*, *view*, *ws* e *html*.

O pacote *html* contém dois pacotes: *parser* e *bean*. O primeiro contém as classes responsáveis por analisar os HTMLs dos sítios e extrair informações específicas. O pacote *bean* contém as classes que representam os elementos do *html*, tais como: formulários, campos de texto, entre outros. Todas as instâncias destas classes são POJOs (*Plain Old Java Object*), ou seja, objetos que possuem somente propriedades e métodos *set* e *get*.

O pacote *control* possui as classes responsáveis por converter os valores para um formato adequado para os objetos responsáveis pela persistência. O pacote *logic* contém

as classes responsáveis por analisar os dados retornados pelas classes do pacote *html*, criar requisições, enviá-las e analisar os resultados.

O pacote *model* contém dois pacotes: *bean* e *pers*. O primeiro contém as classes que representam as entidades do banco de dados. O segundo é formado pelas classes responsáveis por realizar as operações de cadastro, exclusão, atualização e pesquisa no banco de dados.

O pacote *view* é constituído por classes responsáveis pela *interface* administrativa do serviço. Esta *interface* foi desenvolvida em *jsf* e permite que o administrador do serviço defina configurações do *Web Service*, tais como *proxy* e número máximo de *threads* de requisições simultâneas permitidas; e cadastre e atualize informações sobre sítios de *e-commerce* nos quais as buscas poderão ser realizadas. As classes responsáveis pela regra de negócios da *interface* administrativa estão nos pacotes *control* e *bean*.

A Figura 1 mostra os pacotes explicados anteriormente que compõem a arquitetura do *Web Service* desenvolvido.

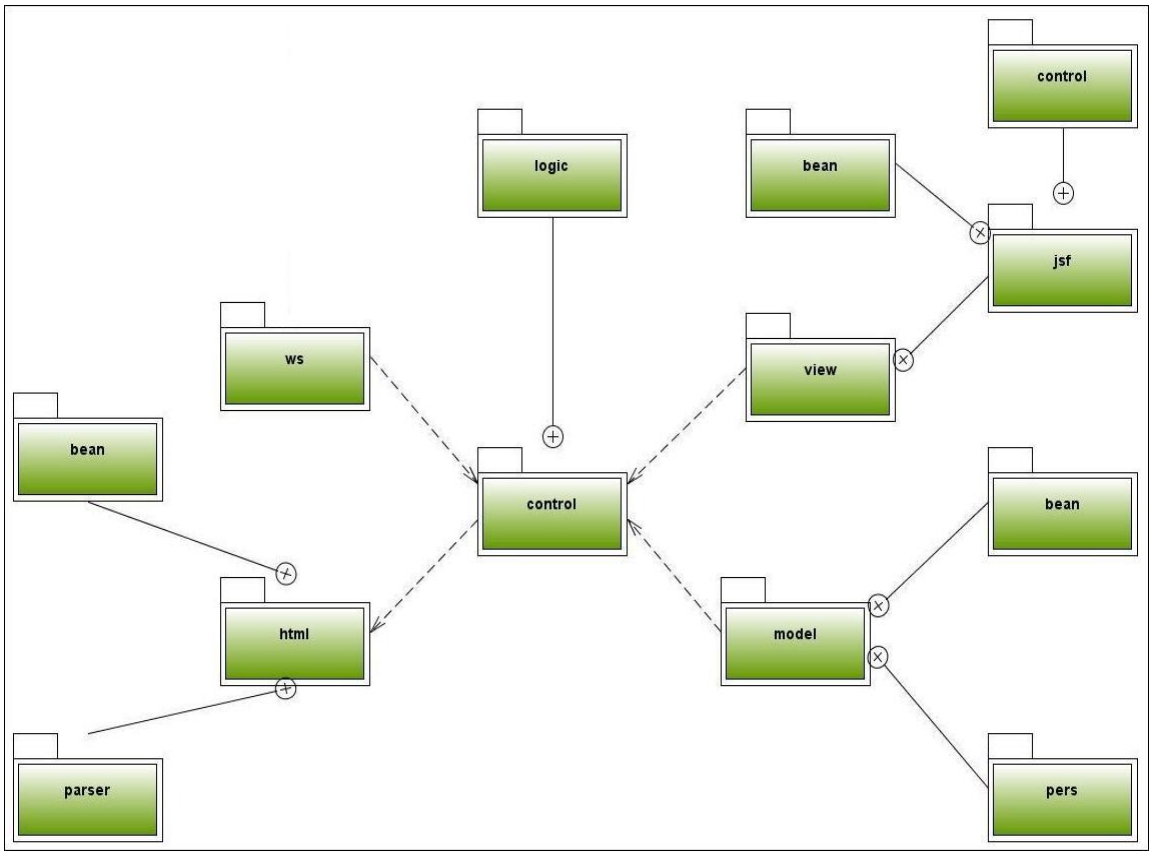

Figura 1. Diagrama de Pacotes do *Web Service* desenvolvido

A Figura 2 apresenta um exemplo de conteúdo de um dos pacotes, a saber, o *ws*. Este é formado pela classe *WsInt*, localizada no pacote *ws*. É a partir dela que é gerada a *interface* de serviço e todos os seus métodos passam a estar disponíveis para os clientes.

Os métodos *getProductPrice*, *getProductPriceFromOneSite* e *getSites* são disponibilizados para os clientes que irão consumir o *Web Service*. O primeiro método realiza uma busca pelos preços do produto em todos os sítios cadastrados no banco de

dados. O segundo realiza uma busca pelos preços do produto em um único sítio, que é passado como parâmetro para o método junto com o nome do produto. Um ponto a ser considerado sobre este método é o de que o *Web Service* aceita somente sítios cadastrados no banco de dados. Assim sendo, é disponibilizado o método *getSites*, que retorna todos os sítios cadastrados no banco dados. Desta forma, o cliente tem como descobrir quais são os sítios cadastrados e pode realizar uma busca em um sítio específico e ignorar os demais.

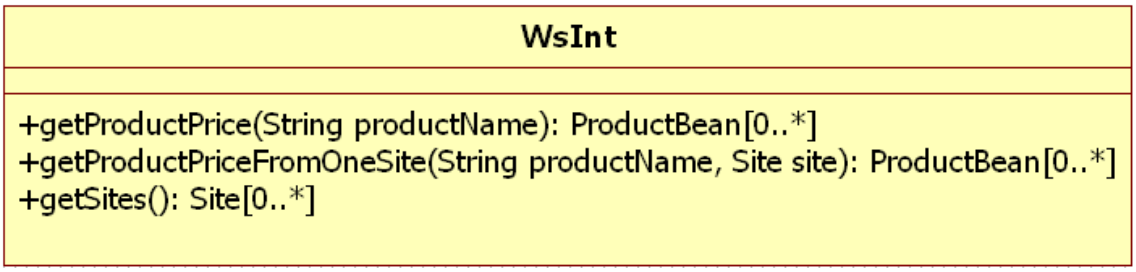

Figura 2. Classe *WsInt*

4.4. Funcionamento do *Web Service*

Para apresentar os resultados obtidos com o uso do *Web Service* foi criada uma aplicação cliente *desktop* para consumir o serviço. A Figura 3 apresenta uma busca por "Impressora" feita através desta aplicação.

Como é possível observar através da Figura 3, o *Web Service* foi capaz de encontrar 51 resultados em 70.125 segundos. A largura de banda de conexão com a internet disponível no momento desta busca era de 512 kbps.

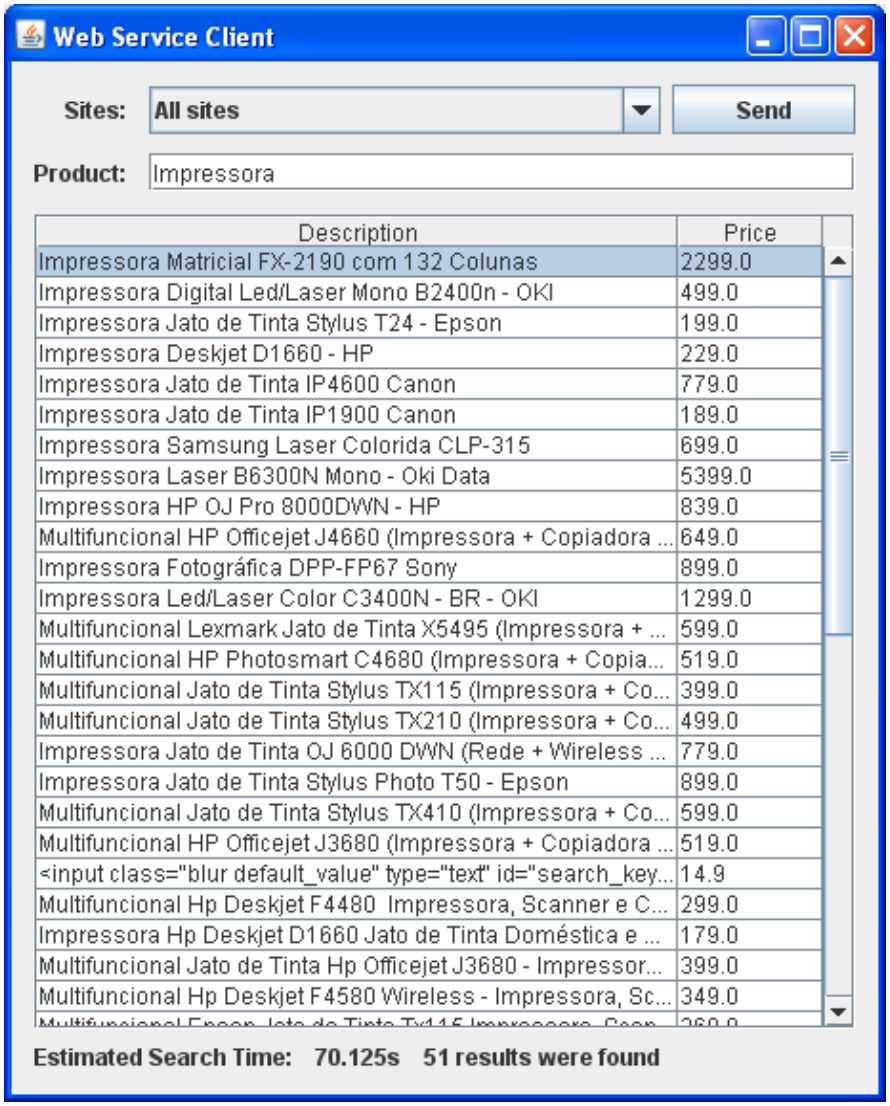

Figura 3. Um Cliente *Desktop* consumindo o Serviço

Em uma das linhas da figura 3 é possível observar que o *web service* retornou um valor inválido, descrito como "*<input class="blur default_value" type="text" id="search_key...*". Este erro ocorreu porque há, em algum lugar da *tag* HTML, a palavra "impressora". Conforme foi explicado na seção 4.1, a associação entre nome do produto e preço é feita baseada em *matchs*. Neste caso, como a aplicação encontrou a palavra "impressora" em algum atributo da *tag*, ela considerou a *tag* como sendo a descrição do produto. A solução para este tipo de falha dependeria de aplicar técnicas de inteligência artificial para, de alguma forma, interpretar o conteúdo e concluir se, de fato, trata-se da descrição do produto. Entretanto, estas técnicas fogem ao escopo deste trabalho.

A largura de banda da conexão com a internet reflete no desempenho do sistema. Quanto maior a largura de banda disponível, melhor será o tempo de resposta do serviço. Evidentemente, outros fatores, como o *hardware* do servidor que hospeda o serviço, também refletem no desempenho do sistema.

5. Conclusões

O *Web Service* proposto é capaz de encontrar os preços de produtos a partir dos sítios de *e-commerce* cadastrados na base de dados. Desta forma, ele supri as necessidades do GPES na implementação do método baseado nas decisões das empresas concorrentes.

Este *Web Service* contém código aberto e disponibiliza uma *interface* para acesso automatizado, diferente de sítios de comparação de preços tradicionais, tais como JaCotei [JACOTEI, 2010] e BuscaPe [BUSCAPE, 2010].

Quanto maior for a banda de conexão com a internet disponível para o serviço, menor será o tempo de busca do *Web Service* proposto para encontrar e retornar os resultados ao usuário.

Ele também possibilita que sejam definidas configurações de *proxy* e número máximo de *threads* no *pool*. Estas configurações são importantes para que o serviço possa ser utilizado dentro de uma rede em que exista um servidor *proxy* e para um melhor desempenho de busca, respectivamente.

Há algumas restrições no serviço desenvolvido, como a incapacidade de realizar buscas em sítios que utilizem o método POST do HTTP ou façam suas buscas baseadas em funções *javascript*.

O *Web Service*, o Cliente *Desktop*, e os respectivos códigos-fonte, podem ser baixados do site do GPES.

5.1. Propostas para trabalhos futuros

A seguir são apresentadas algumas sugestões para trabalhos futuros:

- Uma análise de desempenho: esta análise permitiria mapear quais aspectos influenciam o desempenho do *Web Service*, viabilizando um estudo para otimização do serviço;

- Refatoração do código e aplicação de padrões de projeto: há diversas funcionalidades que podem desenvolvidas para o *Web Service*. Entretanto, para facilitar a manutenção do código e a expansão para acomodar novas funcionalidades, é interessante realizar o *refactoring* do código;

- Aperfeiçoamento da lógica para encontrar os preços de um produto: apesar do *Web Service* encontrar as descrições dos produtos e seus preços com uma taxa de acertos bastante aceitável, é interessante aperfeiçoar a lógica empregada para realizar as buscas utilizando técnicas de inteligência artificial;

- Aplicação de mecanismos de segurança: o *Web Service* não realiza autenticação de clientes e nem envia mensagens criptografadas. O GPES não precisa que estes mecanismos sejam implementados, porém dependendo do contexto no qual o serviço for ser utilizado, a implementação destes mecanismos pode ser crucial;

- Criação de novas funcionalidades: há diversas funcionalidades que podem ser acrescidas ao *Web Service* desenvolvido. Entre elas estão: encontrar, além dos preços, as condições de pagamento do produto; interpretar algumas funções *javascript* e manter um histórico das buscas realizadas e seus resultados.

Referências

Abinader, J.; Lins, R. (2006) “Web Services em Java”. Brasport.

BuscaPe. (2010) Disponível em http://buscape.com.br. Acesso em set-2010.

Cerami, E. (2002) “Web Services Essentials Distributed Applications with XML-RPC, SOAP, UDDI and WSDL”. O’Reilly.

GPES. (2010) Disponível em http://200.134.81.19:8080/gpes. Acesso em set-2010.

JaCotei. (2010) Disponível em http://jacotei.com.br. Acesso em set-2010.

JAX-WS. (2010) Disponível em http://jax-ws.dev.java.net. Acesso em set-2010.

JSF. (2010) Disponível em http://java.sun.com/javaee/javaserverfaces. Acesso em set-2010.

Nakagawa, M. (1995) “ABC: Custeio baseado em atividades”. São Paulo: Atlas.

Nascimento, D.; Vartanian, G. (1999) “O método pleno: Uma abordagem conceitual”. Revista CRC-SP, nº 9.

Oliveira, R.; Crema, R. (2009) “Definição dos pontos de estabilidade, em nível de requisitos, no domínio de preço de venda”. Trabalho de Conclusão de Curso (Graduação em Análise e Desenvolvimento de Sistemas). Universidade Tecnológica Federal do Paraná, Ponta Grossa.

Santos, J. (1991) “Formação de preços e do lucro”. 3ª Ed. São Paulo: Atlas.

Sardinha, J. (1995) “Formação de Preços: A Arte do Negócio”. São Paulo: Makron Books.

SEBRAEPR. (2010) Disponível em http://sebraepr.com.br. Acesso em set-2010.

WSIT. (2010) Disponível em http://wsit.dev.java.net. Acesso em set-2010.