

UTILIZANDO WEB SERVICE PARA INTEGRAÇÃO DE SISTEMAS UM ESTUDO DE CASO: PREFEITURA DE JUIZ DE FORA

Rodolfo Fernandes Peracci

Centro de Ensino Superior de Juiz de Fora, Juiz de Fora, MG

Geraldo Magela Almeida Bessa

Resumo: Este artigo tem como objetivo apresentar a tecnologia Web Services, abordando suas capacidades e benefícios juntamente com sua importância e popularidade. Com o aumento da necessidade de se integrar sistemas, criou-se os serviços web, em que sua principal vantagem é a praticidade e a segurança na hora de fazer transações web. Será demonstrado, aqui, as tecnologias envolvidas. Dentre elas, serão abordados os conceitos de *Web Service*, SOA, SOAP, XML, DTD, WSDL, UDDI e REST. Será abordado um comparativo de forma conceitual entre a tecnologia em foco SOAP e uma alternativa para integração, REST. Ao fim um estudo de caso implementado na Prefeitura de Juiz de Fora onde é demonstrado a utilização de um serviço Web.

Palavras-chave: *Web Services*, integração de sistemas e SOAP.

1 INTRODUÇÃO

Os serviços Web surgiram com o intuito de facilitar a interligação/integração de sistemas, auxiliando na troca de informações entre diferentes aplicações, efetuando uma comunicação.

A utilização da tecnologia de Web Services vem crescendo nos últimos anos devido ao aumento da confiança do consumidor para realizar transações pela internet, sendo viável para as empresas utilizarem cada vez mais essa tecnologia.

Os *Web Services* estão sendo bastante utilizados em aplicações de comércio eletrônico ou, do inglês, *e-commerce*, sendo um exemplo a integração do serviço dos correios com o serviço do comércio eletrônico. O comércio eletrônico solicita ao usuário que informe o número do CEP, enquanto o serviço dos correios se encarrega de retornar o respectivo endereço.

Os *Web Services* fazem parte de soluções utilizadas na integração de sistemas e na comunicação entre aplicações de arquiteturas ou linguagens diferentes. Este trabalho, por sua vez, apresenta de forma conceitual uma comparação da ferramenta em foco SOAP com REST, uma tecnologia alternativa para integração de sistemas. Ao final do artigo, será abordado um estudo de caso realizado na Prefeitura de Juiz de Fora de forma conceitual, em que será demonstrado como funcionam os processos ao decorrer da integração entre os dois sistemas.

Este trabalho terá como forma de definição as aplicações dos *Web Services* em diversas situações na sua utilização, como demonstrado na seção 1 e, em seguida, serão abordados, no item 2, os conceitos de SOA (Arquitetura Orientada a Serviço), uma arquitetura utilizada no desenvolvimento dos *Web Services*. No item 3 será apresentada a comunicação entre camadas, contendo os conceitos de XML, SOAP, WSDL, UDDI e DTD, os quais são responsáveis por efetuar a troca de informações em ambientes distribuídos e descentralizados. Em sequência, no item 4 haverá uma introdução sobre REST, mostrando seus conceitos e um comparativo entre SOAP e REST, demonstrando suas vantagens e desvantagens, enquanto no item 5 será apresentado um estudo de caso utilizando *Web Services* em um sistema da Prefeitura de Juiz de Fora. Por fim, teremos, no item 6, as conclusões deste trabalho.

1.1 APLICAÇÕES PARA WEB SERVICES

Devido à facilidade de implementação e disponibilização, observa-se cada vez mais o aumento no número de possibilidades para aplicações dos serviços web para a integração de sistemas. Segundo Leandro (2005), o *Web Service* tem sido usado em diversos tipos de aplicações, podemos citar algumas como: Utilizar os serviços Web em interfaces de sistemas legados ou *desktop* devido à necessidade de tornar os processos de negócio mais acessíveis.

Com a utilização de *Web Service*, tornam-se desnecessários os gastos com novas infraestruturas de aplicações, havendo, apenas, a preocupação com a criação de interfaces de comunicação para disponibilizar soluções já existentes. Os *Web Services* não estão apenas sendo utilizados em aplicações finais, mas também em tecnologias como *Data Warehouses* ou *Grids* (grades). Existem serviços considerados simples - de uma empresa de turismo, por exemplo -, podem ser combinados para trabalhar com serviços que tenham um nível mais alto de complexidade - um serviço fornecido por hotéis, por exemplo.

Existe uma automatização no gerenciamento de processos de negócio, utilizando uma orquestração de serviços para definir as sequências e condições que um serviço Web requisitará o outro serviço para, então, poder registrar uma funcionalidade. Os processos são padronizados em um tipo de camada de serviços e, dessa forma, eles poderão ser facilmente compostos e associados a um outro serviço, fazendo o que se chama de fluxo de camada de serviços.

1.2 VANTAGENS NA UTILIZAÇÃO DO *WEB SERVICE*

O sucesso da aceitação do *Web Service* no mercado tecnológico vem pelas suas inúmeras vantagens. Segundo Miranda (2005), podem ser destacados, dentre os benefícios existentes a independência de plataforma e linguagem de programação, permitindo que programas de diferentes linguagens e em diferentes plataformas se comuniquem um com o outro de uma forma padrão. Os *Web services* diminuem os custos com a automação entre os processos de negócio, havendo uma queda no custo de transação, além de ser minimizada a possibilidade de ocorrer um erro humano. Contudo, o *Web Service* conta com sua reutilização, podendo facilmente realizar melhorias nas suas funcionalidades e nos conteúdos já existentes, reduzindo, assim, o custo do desenvolvimento. Utilização de protocolos padrão da web, que são: XML, HTTP e TCP/IP.

Muitas empresas se empenham no desenvolvimento de serviços web com uma equipe de pessoas qualificadas para manter essas tecnologias, além de o custo da adoção do *Web Service* em uma empresa ser significativamente menor do que seria caso fossem adotadas as antigas técnicas de integração de sistemas. Flexível, escalável e reutilizável, o *Web Service* pode ser utilizado com intuito de facilitar a localização e implementação dos potenciais utilizadores.

Com a disponibilização dos serviços se torna possível obter novas oportunidades, colocando as funcionalidades das aplicações com o próprio *Web Service*. Pode haver uma procura imediata – como mais serviços de cotação online – ou até indiretamente – como serviços de rastreio de uma encomenda.

2 SOA (*SERVICE-ORIENTED ARCHITETURE*)

Os serviços web foram desenvolvidos a partir da Arquitetura Orientada a Serviço. São funções de negócio, implementadas em software, utilizando como forma de acesso sua interface. Consistem em reorganizar um conjunto de aplicações já em pleno funcionamento e dar suporte a uma infraestrutura em serviços que estão interligados, podendo ser acessados por meio de interfaces padrões ou protocolos de mensagens.

De acordo com Papazoglou (2003), o sucesso dessa arquitetura se dá pelo fato de ser utilizada em múltiplas aplicações, ou seja, ela pode ser executada em um ambiente

onde exista tecnologia e plataformas distintas que necessitam se comunicar.

Segundo Leandro (2005), tanto o *Web Service* quanto a SOA possuem os seguintes conceitos: objetos distribuídos, *middleware* orientado a mensagem e os componentes de software. A noção de contexto e conexão que são baseados nos *brokers* junto da obrigatoriedade do uso de interfaces foi adotado dos objetos distribuídos; já a noção de *middleware* veio por meio da utilização de mensagens trocadas e também do uso das filas. Com relação aos componentes de Software, pode-se dizer que eles vieram dos conceitos do encapsulamento e polimorfismo. Leandro (2005) ainda afirma que existem três papéis básicos nos sistemas que utilizam SOA: há os provedores, os consumidores e os *brokers* (intermediador, no caso), como mostra a figura 1. O provedor torna o serviço disponível e público por meio de um contrato que descreve a sua interface utilizando um registro no *broker*; o consumidor tem o papel de pegar o *broker* que o serviço deseja; já o *broker* fornece ao consumidor a localização e o contrato do serviço solicitado, os quais são utilizados para fazer a conexão (*Binding*).

FIGURA 1
Arquitetura SOA e protocolos *Web Services*



Fonte:Leandro(2005)

Para que todas essas funções sejam utilizadas, um sistema SOA deve prover de três componentes de arquitetura: Transporte para a comunicação de um serviço, o transporte representa os formatos e os protocolos que serão utilizados. A descrição representa os tipos de linguagens que serão utilizados para descrever um serviço,

contendo as informações necessárias para que tal serviço possa ser acessado, quais operações serão realizadas e quais os parâmetros a serem seguidos e o descobrimento, que é o modo que será registrado, anunciado e encontrado o serviço com suas descrições.

3 COMUNICAÇÃO ENTRE CAMADAS

Para se ter a comunicação em aplicações, são utilizadas quatro camadas de empacotamento de requisição e resposta (do inglês, *request and response*) e fazendo uma ligação entre o cliente e o servidor. São elas: XML, SOAP, WSDL e UDDI.

3.1 EXTENSIBLE MARKUP LANGUAGE (XML)

O XML é utilizado para o transporte da informação de forma simples, fácil e ágil, sendo compatível com qualquer plataforma e ficando responsável pela maior parte da interoperabilidade entre as aplicações em um ambiente heterogêneo.

Deitel (2003) define o XML como uma tecnologia aberta com amplitude para aguentar a troca de dados, utilizando como principal característica o uso de marcas (*tags*), permitindo que o desenvolvedor possa configurar o jeito que suas informações serão estruturadas. O XML possibilita a criação de documentos em texto puro e em praticamente todas as formas estruturadas pelo fato de ter o suporte aos formatos *Unicode* e *ASCII*. Assim, tendo essas facilidades de estruturação, o XML é mais leve que o HTML (*Hiper Text Markup Language*).

Ainda de acordo com Deitel (2003), ao contrário do HTML, os documentos podem ser lidos pelos usuários, mas não possuem uma otimização para que sejam tratados pelo computador. Existe uma relação ao armazenamento de dados que são otimizados para manipulação pelos computadores e não para a visualização do usuário, como é o caso do XML.

O autor Tamae (2004) apresenta como vantagens sobre a utilização do XML os seguintes itens:

- Semântica das informações que serão transportadas;
- Facilidade de visualizar os diferentes tipos de dados;

- Compartilhamento de dados de forma rápida entre as aplicações;
- Independência de plataforma.

O autor Tamae (2004) exemplifica o benefício que o XML oferece quando o usuário tem necessidade de fazer uma busca de um determinado assunto, como localizar um hotel na Ilha de Páscoa. Se essa busca fosse realizada pelas formas tradicionais que temos, o resultado seria ineficaz, uma vez que o usuário provavelmente receberia uma lista de links relacionados ao dia da Páscoa e não sobre a Ilha de Páscoa.

Segundo Siverschatz et al. (2006), a linguagem XML tem como característica ser autodescritiva, pelo fato de o XML utilizar as marcas (*tags*), a mensagem é facilmente interpretada, tornando-a autoexplicativa, sem a necessidade de consultar um esquema para entender o significado do texto. Outra característica para é ser flexível, pelo fato de o XML ser estruturado, não seguir uma estrutura rígida, o que permite que as *tags* especifiquem atributos de um elemento que apareça em algumas partes do documento e em outras não.

O XML como a maior parte dos serviços web é de padrão aberto – Independente de plataforma e sistema operacionais, pode ser utilizado na integração também de sistemas heterogêneos. E extensível, a criação de *tags* de um modo arbitrário permite ao documento XML ser adaptado a praticamente todo o domínio de problema.

3.1.1 ESTRUTURA DO XML

O XML é definido basicamente por elementos, dados e (opcionalmente) atributos das *tags*. Um elemento é composto por pares de tags que informam o início e o fim com seu respectivo texto contido nas *tags*. De acordo com Siverschatz et al. (2006), é necessário que o documento XML tenha um único elemento raiz, o qual irá compreender todos os elementos do documento, conforme a figura 2.

FIGURA 2
Exemplo estrutura de um XML

```
1  <?xml version="1.0"?>
2  <pedido_medicação>
3      <id> "PedidoMedicacao_1" </id>
4      <solicitante>
5          <nome> "Maria Joana da Silva" </nome>
6          <unidade> "UAPS Progresso" </unidade>
7      </solicitante>
8      <lista_medicamento>
9          <item>
10             <id> "123456" </id>
11             <medicamento>"Captopril 25 MG Comprimido" </medicamento>
12             <quantidade> 1000 </quantidade>
13         </item>
14         <item>
15             <id> "7891011" </id>
16             <medicamento>"Dipirona 100 MG Comprimido" </medicamento>
17             <quantidade> 1500 </quantidade>
18         </item>
19     </lista_medicamento>
20 </pedido_medicação>
```

Fonte: Prefeitura de Juiz de Fora (2013)

Conforme Siverschatz et al. (2006), existe a forma correta e a forma errada de o texto aparecer no contexto de um elemento:

- Exemplo de alinhamento correto: <item> <id> </id> </item>
- Exemplo de alinhamento errado: <item> <id> </item> </id>

Ainda de acordo com Siverschatz et al. (2006), os atributos são usados na descrição dos elementos ou para adicionar outras informações. Eles aparecem em pares valor=real e, em seguida, o fechamento da *tag* (>).

3.1.2 ESQUEMA DO DOCUMENTO XML

Segundo Vargas (2008), o esquema XML (*XML Schema*) é utilizado para descrever uma estrutura para o documento XML, pois esse documento pode conter inúmeros elementos, incluindo elementos aninhados. Para isso, há uma necessidade de validar o XML a fim de verificar se as regras da linguagem e documento estão estruturadas conforme acordado entre as partes. Sendo assim, é possível ser processado automaticamente como parte de uma aplicação.

3.2 DTD (*DOCUMENT TYPE DEFINITION*)

Quando uma aplicação envia dados que não estão estruturados, os DTDs ajudam a validar esses dados que estão sendo recebidos. DTDs são opcionais e os dados enviados com um DTD são chamados de dados XML Válidos.

De acordo com Silberschatz et al. (2006), os DTDs consistem em restringir as informações e os tipos de informações em um documento XML. Em resumo, o DTD escreve a estrutura do documento, não restringindo o tipo de dados como inteiro, cadeia de caracteres ou real; ele apenas restringe o aparecimento de subelementos e atributos.

O DTD contém informações sobre as *tags* que existem no documento XML, que correspondem ao DTD, abrangendo as regras que definem os padrões de subelementos. Assim, as aplicações que utilizam XML para intercambiar os dados podem validar seus documentos antes de serem processados através do DTD.

Macoratti (2000) define a estrutura básica de uma DTD como:

- Composta por declarações: `<! >`
- Definição de elemento: `<!ELEMENT . . . >`
- Definição de atributos: `<!ATTLIST . . . >`
- Definição de entidades: `<!ENTITY . . . >`
- Definição de novos tipos: `<!NOTATION . . . >`

Quando um documento XML não utiliza DTD, são caracterizando como um documento bem formado, existindo algumas restrições em suas marcações como: em um atributo não pode ser atribuído um valor *Default*, todos os atributos são considerados CDATA e também OPCIONAIS e as entidades não podem ser definidas.

3.3 SOAP (SIMPLE OBJECT ACCESS PROTOCOL)

SOAP é um protocolo utilizado para troca de mensagens em ambiente independente de plataforma e linguagem de programação, tendo suporte para qualquer método de codificação de dados. O SOAP pode ser utilizado em combinação com uma variedade de outros protocolos, como HTTP, SMTP, FTP dentre outros.

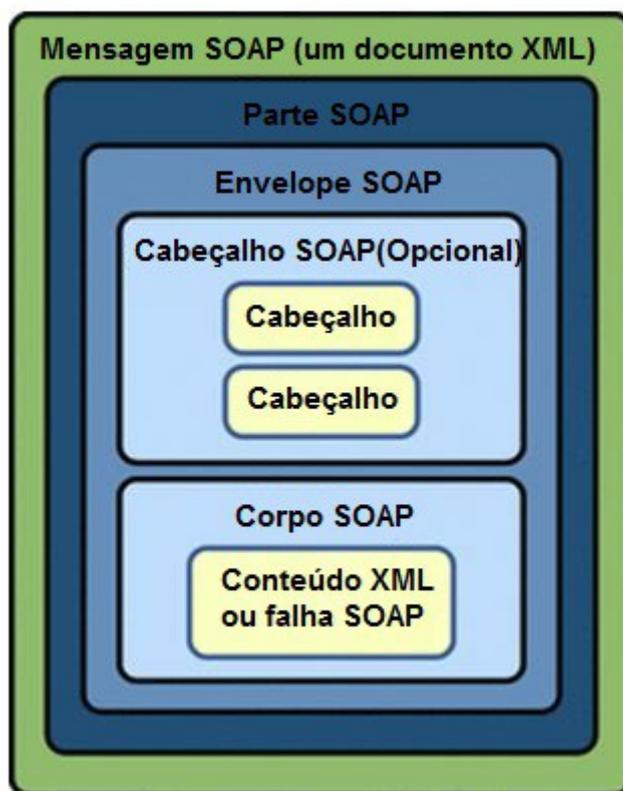
Segundo Deitel (2003), o SOAP possibilita as aplicações de se comunicarem de forma fácil e ágil pela internet, tendo como ferramenta o XML (chamado de mensagens do SOAP). Para a definição de sua estrutura de comunicação, o modelo de objeto já inclui todos os tipos de funções e suas capacidades.

De acordo com Poleza (2007), em relação à transferência de mensagens, o SOAP define um *framework* para efetuar a troca de dados via XML, em que esse *framework* é independente de qualquer obstáculo operacional que possa ocorrer. Muschamp (2004), define que o protocolo SOAP abrange as seguintes partes: A mensagem SOAP contém um elemento chamado envelope. Tal elemento configura um *framework* para escrever o conteúdo da mensagem e para processá-la.

Contém uma série de regras de codificação que envia instâncias de tipo de dados dentro de uma mensagem e utiliza tipos de convenções para definir a maneira pela qual os métodos devem ser chamados e suas respectivas respostas, com isso é feito uma conexão para trocar mensagens usando protocolos de comunicação. SOAP possui a característica de utilizar o padrão *request - response*, em que um componente solicita a outro componente alguma coisa e onde esse segundo componente pode ou não entregar a resposta. O envio das mensagens é feito via HTTP *request* e o recebimento de uma resposta é feita via *response*.

Segundo Zarelli (2012), uma mensagem SOAP é nada mais que um XML comum que contém um elemento que é chamado de envelope, o qual identifica o XML como uma mensagem SOAP. Um envelope é composto pelo Cabeçalho e o Corpo. Observe a figura 3:

FIGURA 3
Mensagem SOAP



Fonte: Traduzido de Zarelli (2012)

Ainda de acordo com Zarelli (2012), o envelope é o elemento raiz do documento XML. Ele possui dois tipos de atributos, sendo eles os *xmlns:soap* e o *soap:encodingStyle*. O primeiro define o *namespaces*, já o segundo define como os dados serão representados no documento XML, ou seja, o tipo de dados que será utilizado no documento.

De acordo com Dantas (2007) e Zarelli (2012), o Cabeçalho é opcional, que carrega informações adicionais e específicas do aplicativo da mensagem SOAP. Dantas (2007) afirma que durante o tráfego pela rede a mensagem normalmente caminha por vários "nós" intermediários até conseguir chegar ao seu destino. Por essa razão, o Cabeçalho deve estar presente no elemento filho do envelope, conforme a figura 3.

Corpo é um elemento obrigatório e contém o *payload*, ou seja, as informações que serão transportadas para seu destino final. Dentro do Corpo contém o elemento opcional Falha, conforme a figura 4, que é utilizado para carregar informações de status ou erros retornados pelos "nós".

FIGURA 4
Esqueleto da mensagem SOAP

```
1 <?xml version="1.0"?>
2 <soap:Envelope
3 xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
4 soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
5
6 <soap:Header>
7 ...
8 </soap:Header>
9
10 <soap:Body>
11 ...
12 <soap:Fault>
13 ...
14 </soap:Fault>
15 </soap:Body>
16
17 </soap:Envelope>
```

Fonte: Zarelli (2012)

3.4 WSDL (WEBSERVICES DESCRIPTION LANGUAGE)

O WSDL foi criado para descrever e publicar os padrões e protocolos de um *web service*, estabelecendo um modelo comum para sua descrição. Essa tecnologia permite que haja uma divisão das funcionalidades abstratas oferecidas pelo serviço dos detalhes concretos do mesmo, como por exemplo de que forma e onde as funcionalidades do serviço são oferecidos. Christensent (2001) afirma que o WSDL define uma forma padronizada para que todos os detalhes de um serviço web sejam adequadamente especificados.

O WSDL possui vários componentes que descrevem seus serviços. Segundo NewComer (2002), utilização de dados por meio dos esquemas XML ou outros mecanismos são definidos os tipos de dados a serem utilizados nas mensagens. As mensagens são definições abstratas dos dados, seja de um documento inteiro ou para o mapeamento de um novo método de inovação.

As operações também são definidas de forma abstrata como fila de mensagem, nomeação de um método ou processos comerciais em que será ou não aceita e processada a mensagem, o *Port Type* é definido como um conjunto de operações, os formatos de dados e protocolos concretos para as operações e mensagens que foram definidas para uma determinada *Port Type*, se aplicam na *Binding*.

A porta é a unificação do endereço de rede com uma *binding*, fornecendo, assim, o endereço para o serviço de comunicação e por fim o *Service*, uma coleção de port type.

3.4.1 DOCUMENTO WSDL

Segundo Leandro (2005), um documento WSDL completo pode ser dividido em 3 partes. A primeira parte descreve o *Service type* (tipo de serviço) e conterá basicamente o que o serviço fará. Essa interface abstrata é responsável por definir uma interface lógica que contém um conjunto de operações que o serviço realiza. Nessas operações são definidos: as mensagens de entrada e de saída, o formato de cada mensagem, os tipos de dados que cada elemento possuirá na mensagem.

A segunda parte define o *binding* (ligação) das interfaces abstrata com um conjunto concreto de protocolos. O *binding* indica o estilo de codificação, especifica se um esquema XML será necessário, qual o protocolo é o mais indicado para construir o envelope, quais os cabeçalhos serão inclusos na mensagem e qual protocolo de transferência deve ser usado.

A terceira e última parte do documento WSDL descreve a implementação de um serviço, tratando-se de uma coleção de *ports* relacionados que pode ser apenas um ou até mais de um. Cada *port* implementa uma *binding* indicando o ponto de acesso a um *end point* (ponto final) de um serviço. O provedor de serviço pode distribuir vários pontos de acesso ao mesmo serviço, porém cada ponto deve estar implementado a uma *binding* diferente.

3.5 UDDI (UNIVERSAL DESCRIPTION, DISCOVERY AND INTEGRATION)

O UDDI é responsável pela descoberta dos serviços Web. Nele é fornecido um diretório público ou privado (centralizado) para que seja disponibilizado ao usuário os serviços desejados. Atualmente os diretórios privados são utilizados em processos internos de empresas.

Segundo Leandro (2005), as especificações UDDI definem os registros para o serviço, o qual é acessado através de mensagens SOAP. Essas especificações possibilitam que as informações sejam gerenciadas sobre tipos de serviços, implementações de serviços e provedores. Esse mecanismo é utilizado para categorizar, descobrir e realizar ligações a serviços. Para anunciar os serviços oferecidos, são usados os provedores. Já os consumidores são utilizados para localizar serviços que necessitam e obter os metadados necessários para consumi-los. É importante salientar que a utilização do UDDI não é obrigatória. Se o consumidor já obtiver as informações

sobre o serviço que quer acessar, não há necessidade de pesquisá-lo, podendo usar o modo mais direto: o *Web Services Metadata Exchange*.

4 INTRODUÇÃO A REST (REPRESENTATIONAL STATE TRANSFER), UM SERVIÇO ALTERNATIVO PARA O SOAP.

Segundo Rozlog (2013), para uma outra opção de integração de sistema, os desenvolvedores têm exposto seus serviços fazendo a utilização do REST usando o padrão de URI (*Uniform Resource Identifier*). Assim, utilizam como exemplo a seguinte chamada de um serviço web: "www.meuwebservice.com.br/webservice/metodo?Parâmetro=xxxx".

REST é utilizado nos serviços web como uma arquitetura, tendo como o RESTful o serviço que faz o uso de seu paradigma.

Diferente do SOAP, o REST conta com aplicativos Web fracamente acoplados, os quais utilizam recursos localizadores como URL, URI e URN (vale lembrar que não é utilizado mensagem). REST utiliza, como transporte bem-sucedido pela Web, o HTTP, ou seja, ele alavanca aspectos do protocolo HTTP com pedidos *GET, POST, PUT, DELETE e HEAD*, em que esses pedidos podem ser facilmente mapeados de acordo com a necessidade da aplicação de negócios padrão.

4.1 VANTAGENS E DESVANTAGENS REST

Segundo Cavalcanti (2011), pode-se citar, como vantagens e desvantagens no uso da arquitetura REST, os seguintes aspectos:

Vantagens

- Pode-se utilizar N camadas intermediárias na aplicação, como: *proxys, gateways, cache servers* para que se consiga ganhar um melhor desempenho e agilidade com segurança;
- É mais seguro do que SOAP, já que existe a possibilidade de usar REST sobre

HTTPS, usar os próprios recursos de segurança padrão HTTP, além do uso de *firewall* e outros recursos;

- É utilizado para fazer pedidos (dados) *GET*, *POST*, *PUT*, *DELETE* e *HEAD*; já o SOAP, apenas o *POST*.

Desvantagens

- O REST não utiliza uma interface para que possam ser definidos seus serviços;
- REST utiliza apenas o padrão WEB (HTTP) para que possa haver a sua troca de dados, em que se pode ter uma perda de segurança;
- Até hoje, não existe uma ferramenta que apoie o desenvolvimento dos serviços do REST;
- Não permite requisições assíncronas.

5 COMPARTATIVO SOAP VS REST

Lima (2012) menciona as vantagens e desvantagens do SOAP em relação ao REST como mostra a tabela 1:

Tabela 1
Comparativo SOAP sobre REST

SOAP vs Rest	Vantagens	Desvantagens
Especificações	Todos os protocolos do SOAP são melhores especificados	Criar interfaces WSDL manualmente pode ser trabalhoso, entretanto, apesar de existirem ferramentas que as geram automaticamente, sua utilização pode tornar o WSDL um tanto frágil com risco de falhas.

Transporte	Independência de transporte: as mensagens SOAP são independentes de protocolo para que sejam transportadas. O envelope pode ser transportado em quaisquer protocolos, sejam HTTPS, SMTP, TCP e etc;	-
	Os dados do SOAP são estruturados usando XML. Portanto, as mensagens podem ser compreendidas por quase todas as plataformas de hardware, sistemas operacionais e linguagens de programação.	
Mapeamento	SOAP faz o mapeamento satisfatoriamente para o padrão de solicitação/resposta HTTP	Não pode ser utilizado em Cache, pois SOAP utiliza apenas métodos HTTP POST
Segurança	Pode ser usado tanto de maneira anônima quanto por meio de autenticação, nome e senha;	-
	Os protocolos de segurança são melhor especificados e difundidos em SOAP (Web ServicesDL, Web Services-Security, Web Services-Addressing, Web Services-Coordination, Web Services-ReliableMessaging) do que HTTP utilizado pelo Rest	-

Fonte: Lima (2012)

6 ESTUDO DE CASO

O cenário de aplicação descrito nesse artigo a fim de demonstrar a utilização de uma infraestrutura de suporte a transações em *Web Service* será de um sistema da Prefeitura de Juiz de Fora, DIM (Dispensação Individualizada de Medicamentos), responsável pela dispensação de medicamentos nas UAPS (Unidades de Atenção Primária a Saúde), onde havia a necessidade de uma integração com o sistema de uma empresa terceirizada, que distribui medicamentos para todas as unidades.

Cenário: O sistema DIM implantado em 2012 tem como sua principal função a dispensação de medicamento nas farmácias das unidades em cada bairro, junto com o controle e ajuste de estoque. Mensalmente, é realizado o pedido dos medicamentos pelo sistema da empresa terceirizada e logo são recebidos nessas unidades os medicamentos referentes à solicitação. Com isso, o usuário responsável pela dispensação no sistema tinha a função de, além de dispensar o medicamento, realizar a entrada no estoque do sistema (manualmente, um medicamento de cada vez) de aproximadamente 120 medicamentos. Isso demandava tempo e desgaste do funcionário.

Solução: Diante das dificuldades apresentadas acima, em 2013 a solução encontrada foi a utilização do *Web Service*, para que pudesse haver a comunicação entre os dois sistemas utilizando como referência a tecnologia SOAP, melhorando, assim, a entrada de material no estoque do DIM.

Ferramentas e tecnologias: O DIM foi desenvolvido utilizando como linguagem de programação o PHP. Para o armazenamento de dados, foi utilizado o MySQL.

6.1 INTEGRAÇÃO DIM

Para melhor entendimento de como foi realizada a integração entre os sistemas, resumimos em 5 passos:

1. Primeiramente, a farmacêutica da UAPS acessa o sistema da empresa terceirizada;
2. Em seguida, solicita a reposição dos medicamentos, em que são informados os medicamentos e a quantidade necessária;
3. É feita uma verificação do estoque de medicamentos da empresa terceirizada e a aprovação do pedido feito;

4. Um serviço Web é disparado, contendo as informações solicitadas dos medicamentos para banco de dado e, em seguida, para o estoque do DIM;
5. Por fim, com os medicamentos já no estoque, a farmacêutica pode realizar as dispensações.

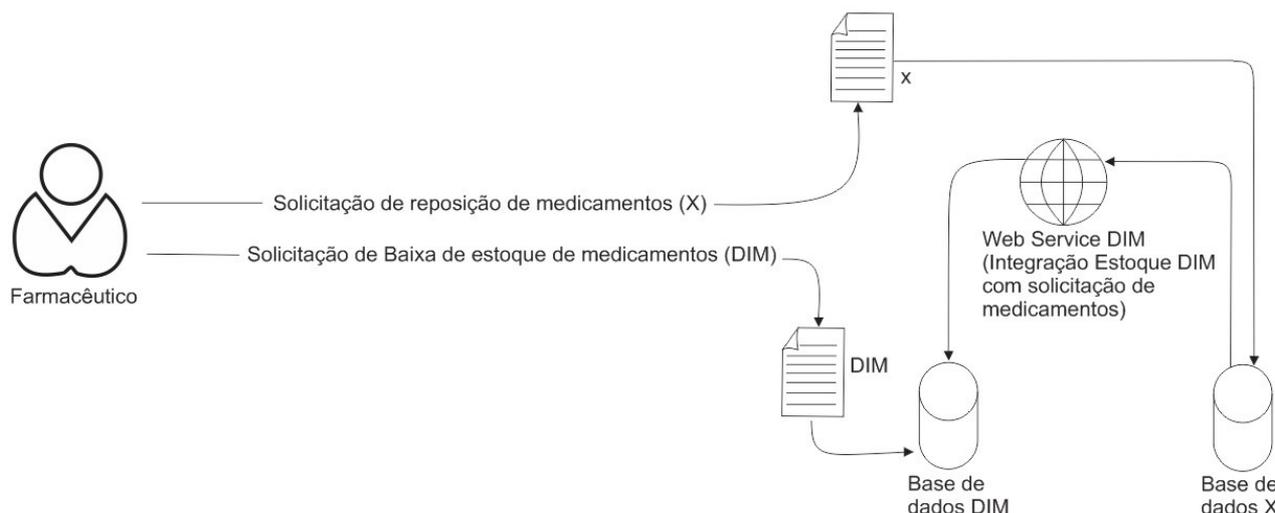
6.2 FLUXOGRAMA

Abaixo segue o fluxograma da integração:

DIM – Sistema responsável pela dispensação de medicamentos nas UAPS.

X – Sistema responsável pela distribuição/ logística dos medicamentos

FIGURA 5
Fluxograma



Fonte: Prefeitura de Juiz de Fora (2013)

Com a utilização do *Web Services* para a integração dos dois sistemas houve uma diminuição do tempo de espera do paciente para entrega do seu medicamento, um menor desgaste para o funcionário que faz a dispensação, aperfeiçoamento do sistema, disponibilização dos serviços para futuras empresas que assumirem a distribuição dos medicamentos e agilidade nos processos.

Algumas dificuldades foram encontradas e solucionadas como: Os medicamentos não poderiam estar disponíveis para a dispensação sem o medicamento estar fisicamente nas UAPS. Para solucionar esse problema, foi adotado o processo de apenas disparar a integração quando o caminhão sair para a entrega dos medicamentos. Apesar de

solucionar essa dificuldade ainda fica alguns pontos fracos: Dependência da utilização correta dos processos pela empresa terceirizada, falhas técnicas podem ocorrer um atraso para o paciente na hora de retirar o medicamento.

No primeiro semestre de 2014 o sistema DIM foi inativo, devia a entrada de uma nova empresa terceirizada para a distribuição de medicamentos, a mesma fornece um sistema próprio para a dispensação de medicamentos.

7 CONCLUSÃO

Diante das duas tecnologias populares e de grande potencial, como demonstrado ao decorrer desse artigo, cuidado deve ser tomado ao apontar qual tecnologia é superior. É normal seguir o modismo e as tendências da época sem ter um prévio conhecimento que justifique a escolha da tecnologia.

O intuito desse trabalho foi demonstrar as vantagens e facilidades adquiridas ao utilizar o *Web Service*, focando a tecnologia SOAP, englobando suas funcionalidades e conceitos. Apresentou-se também, no capítulo 4.1, uma alternativa para o SOAP, que é o REST, o intuito de sua apresentação é demonstrar que existem tecnologias com conceitos diferentes de SOAP porém com a mesma funcionalidade, o qual também possui vantagens e desvantagens sobre a tecnologia em foco.

O cenário apresentado pela Prefeitura de Juiz de Fora demonstra as necessidades e limitações de um sistema (DIM) para alimentar o estoque de medicamentos, e com a implantação de um *Web Service* houve a solução desses problemas, na seção 6.2 é especificado as melhorias adquiridas pela integração.

Como visto nesse artigo, o REST mostra facilidade de implementação, agilidade no serviço e a possibilidade de trabalhar com o protocolo HTTPS, não necessitando acoplar novos protocolos de segurança. Já o SOAP é um padrão que, combinado com as especificações do *Web Service*, pode garantir questões de QoS (*Quality of Service*), segurança, transações e outras questões encontradas em integrações de maior complexidade, uma vez que seus protocolos são melhor especificados em relação ao REST.

Para finalizar pode-se citar que nem sempre a melhor tecnologia empregada dentro de um contexto será a melhor opção para qualquer outro, devendo sempre obter o conhecimento do cenário para que se possa utilizar a tecnologia mais adequada.

REFERÊNCIAS

ABINADER, J. A.; LINS, R. D.. **Web Services em Java**. Brasport, 2006.

CAVALCANTI, Marcos. REST Web Services. Disponível em: <<http://www.docstoc.com/docs/108413189/REST-Web-Services>>. Acessado em 06 de Maio de 2014.

CHINNICI R.; GUDGIN M.; MOREAU, J.; WEERAWARANA, S. **Web Services Description Language (Web ServicesDL) version 1.2. Technical report, W3C, 2003.**

DANTAS, Daniel Chaves Toscano. **Simple Object Access Protocol (SOAP)**. Disponível em: <http://www.gta.ufrj.br/grad/07_2/daniel/>. Acessado em 06 de Maio de 2014.

DEITEL, H.M.; DEITEL, P.J.; NIETO, T. Internet & World Wide Web – Como Programar. 2. Porto Alegre, RS, Brasil: Ed. Bookman, 2003A.

LEANDRO, Sabrina da Silva. **Balanceamento de Carga em Web Services**, 2005. 81 f. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) – Universidade Federal de Santa Catarina, Florianópolis, 2005.

LIMA, Jean Carlos Rosário. **Web Services (SOAP X REST)**. 2012 . 40 f. Monografia apresentada à faculdade de Tecnologia de São Paulo para a obtenção do Grau de Tecnólogo em Processamento de Dados. São Paulo, 2012.

MACORATTI, José Carlos. **XML – Usando DTD – Document Type Definition**. Disponível em <http://www.macoratti.net/vb_xml2.htm>. Acessado em 11 de novembro 2013.

MIRANDA, Leonel. **Conceitos de Web Services**. Disponível em: <http://www.sinfic.pt/SinficNeWeb_Servicesletter/sinfic/NeWeb_Servicesletter43/Dossier3.html>. Acessado em 02 de Abril de 2013

MITRA, Nilo. SOAP Version 1.2 Part 0: Primer. **Apresenta a especificação da W3C para a tecnologia SOAP**. Jun. 2003. Disponível em: < <http://www.w3.org/TR/2003/REC-soap12-part0-20030624/>>. Acessado em 16 novembro de 2013.

NEWCOMER, Eric. **Understanding Web Services: XML, Web ServicesDL, SOAP and UDDI**. Addison-Wesley, 2002.

OLIVEIRA, Marcelo. **REST e RESTful: a diferença**. Disponível em: <<http://marceloweb.info/rest-e-restful-a-diferenca/>>. Acessado em 02 de fevr de 2013.

PAPAZOGLU, Mike P. **Service-Oriented Computing: Concepts, Characteristics and Directions**. In: **Web Information Systems Engineering Workshops**, 2003 (WISE' 03), 04, 2003. Proceedings Fourth *International Conference on Web Information Systems Engineering*. IEEE, 2003. p. 3-12.

ROZLOG, Mike. **REST e SOAP: Usar um dos dois ou ambos?**. Disponível em :<<http://www.infoq.com/br/articles/rest-soap-when-to-use-each>>. Acessado em 06 de Maio de 2014.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. **Sistema de Banco de Dados. São Paulo**: Elsevier Editora, 2006. 781p.

TAMAE, Rodrigo Yoshio, MUZZI, Fernando Augusto Garcia. XML – **Quebrando as barreiras da interoperabilidade entre as plataformas**. FAEG/GARÇA, 2004.

VARGAS, Roberto Silva. **Usando Web Services para acesso à informação de contexto em um portal Web**. Pelotas, 2008.

ZARELLI, *Guilherme Biff*. **Como funciona o SOAP – Protocolo Simples de Acesso a Objetos**. Disponível em:<<http://helpdev.com.br/2012/03/22/como-funciona-o-soap-protocolo-simples-de-acesso-a-objetos/>>. Acessado em 12 de novembro de 2013.