UNIVERSIDADE FEDERAL DE SANTA CATARINA

TECNOLOGIAS DA INFORMAÇÃO E COMUNICAÇÃO

MÁRCIO ELIAS HAHN DO NASCIMENTO

UMA ARQUITETURA DE SERVIÇOS WEB COMO MEIO DE INTERCÂMBIO DE DADOS ENTRE SISTEMAS HETEROGÊNEOS

Araranguá, 25 de Fevereiro de 2013

MÁRCIO ELIAS HAHN DO NASCIMENTO

UMA ARQUITETURA DE SERVIÇOS WEB COMO MEIO DE INTERCÂMBIO DE DADOS ENTRE SISTEMAS HETEROGÊNEOS

Trabalho de Conclusão de Curso submetido à Universidade Federal de Santa Catarina como parte dos requisitos necessários para a obtenção do Grau de Bacharel em Tecnologias da Informação e Comunicação. Sob a orientação do Professor Alexandre Leopoldo Gonçalves.

Araranguá, 2013

Márcio Elias Hahn do Nascimento

UMA ARQUITETURA DE SERVIÇOS WEB COMO MEIO DE INTERCÂMBIO DE DADOS ENTRE SISTEMAS HETEROGÊNEOS

Trabalho de Conclusão de Conclusão de Curso submetido à Universidade Federal de Santa Catarina, como parte dos requisitos necessários para a obtenção do Grau de Bacharel em Tecnologias da Informação e Comunicação.

Professor Alexandre Leopoldo Gonçalves, Dr.
Presidente da Banca - Orientador

Professor Juarez Bento da Silva, Dr.
Membro

Professor Carlos André de Sousa Rocha, MSc.
Membro

Araranguá, 25 de fevereiro de 2013

Este trabalho é dedicado a minha esposa Marciele e meu filho Nicolas, em forma de retratação pelas horas dispendidas na elaboração deste, horas nas quais estive ausente.

AGRADECIMENTOS

Agradeço a meus pais, Licia e Enedir, minha esposa e filho, Marciele e Nicolas, pela compreensão e apoio nos momentos mais críticos desta jornada. A meus colegas pela amizade e a meus professores, em especial ao Prof. Dr. Alexandre Leopoldo Gonçalves, pela orientação e suporte para a realização deste trabalho.

“Se, a princípio, a ideia não é absurda, então não há esperança para ela.”

Albert Einstein

RESUMO

No contexto tecnológico atual, tecnologias emergem cotidianamente e com elas a necessidade de integração visando proporcionar uma maior flexibilidade em qualquer campo atendido por um dado conjunto tecnológico distinto. No âmbito de negócios, informações são consideradas um bem vital para a organização. Por este fato, a centralização das mesmas agiliza e facilita a tomada de decisão, bem como a execução de qualquer processo inerente ao negócio. Diferente do modelo lógico empresarial, onde todas as informações dos diversos setores que compõe a organização são centralizadas, no modelo físico, tais informações estão dispostas em vários sistemas, dispositivos, bases de dados, e outros recursos computacionais, empregando diferentes tecnologias. Com o propósito de interligar de modo transparentes estes componentes heterogêneos surge a necessidade de um meio capaz de atingir tal objetivo. É neste contexto que se torna relevante o emprego da tecnologia de serviços web. A arquitetura de serviços web está em expansão, e tem grande aceitabilidade no meio tecnológico, por proporcionar uma interface padronizada, possibilitando a comunicação entre diferentes plataformas com uma curva de adaptação reduzida. Propõe-se aqui uma arquitetura de software que implementa uma interface de serviços web para promover o intercâmbio de dados entre dispositivos móveis e um sistema ERP. Empregando conceitos de serialização de objetos, protocolo de transporte e servidores de aplicação, a solução tem o propósito de promover a comunicação entre as duas plataformas de modo transparente para os usuários. Busca-se ainda um desempenho satisfatório dentro das restrições de redes móveis e poder de processamento desses dispositivos. A aplicação da arquitetura proposta proporcionou um aumento no desempenho das equipes de vendas a qual utilizam os dispositivos móveis. Em virtude da precariedade das redes móveis, na solução atual, uma simples transferência de um pedido de compras para o sistema ERP da empresa vinculada ao estudo de caso era mensurada em minutos. Com a solução de serviços web, foi possível a redução desta unidade para segundos.

Palavras-chave: Serviços Web, SOAP, RESTful, XML, JSON

ABSTRACT

In the current technological environment new technologies emerge daily and with them the need for integration. It provides flexibility in any field supported by a particular set technological. Within business, information is considered a vital asset to the organization. Thus, the centralization of it speeds up and facilitates the decision-making and the execution of any process inherent to the business. Unlike the logical business model, where all the information from various sectors that make up the organization are centralized, in the physical model such information is arranged in various systems, devices, databases, and other computer resources by using different technologies. In order to interconnect in a transparent these heterogeneous components, it is proposed a solution using web services. The web service architecture is in expansion and has great acceptability by providing a standardized interface and by enabling communication among different platforms with a reduced adaptation curve. In this work is proposed a software architecture that implements a web services interface in order to promote the exchange of data between mobile devices and an ERP system. Employing concepts of object serialization, transport protocol and application servers, the solution is designed to promote transparently the communication between the two platforms. Also, is still intended to get satisfactory performance within the constraints of mobile networks and processing power of today's mobile devices when compared to the solution previously adopted. Once applied the proposed architecture we observed an increase in performance of the sales teams which use mobile devices. Given the precarious nature of mobile networks in the current solution a simple transfer of a purchase order for the company's ERP system was measured in minutes. With the web services solution it was possible to reduce this unit to seconds.

Keywords: Web Services, SOAP, RESTful, XML, JSON

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

3G	Terceira Geração das Tecnologias Móveis
API	*Application Programing Interface*
CDL	*Change Detection by Level*
CORBA	*Comon Object Request Broker Architecture*
CRUD	*Create, Read, Update, Delete*
DCOM	*Distributed Component Object Model*
EDGE	*Enhanced Data-Rates for Global Evolution*
ERP	*Enterprise Resource Planning*
FTP	*File Transfer Protocol*
GPRS	*General Packet Radio Services*
HA	*High Avaiability*
HTML	*Hyper Text Markup Language*
HTTP	*Hyper Text Transfer Protocol*
HTTPS	*Hyper Text Transfer Protocol Secure*
IDE	*Integred Development Enviroment*
IDL	*Interface Definition Language*
JSON	*JavaScript Object Notation*
RDF	*Resource Definition Framework*
REST	*Representation State Transfer*
ROA	*REST Oriented Architecture*
RPC	*Remote Procedure Call*
RSS	*Really Simple Syndication*
SDK	*Software Development Kit*
SGDB	Sistema de Gerenciamento de Banco de Dados
SMTP	*Simple Mail Transfer Protocol*
SOAP	*Simple Object Application Protocol*
SQL	*Strutured Query Language*
SSL	*Secure Socket Layer*
TCP/IP	*Transfer Control Protocol/Internet Protocol*
TIC	Tecnologias da Informação e Comunicação
UDDI	*Universal Description, Discovery and Integration*
Unix	Sistema Operacional
URI	*Unified Resource Identification*
URL	*Unified Resource Locator*

W3C	*World Wide Web Consorcium*
WADL	*Web Application Definition Language*
Web	*World Wide Web* ou *WWW*
WSDL	*Web Service Description Language*
XHTML	*eXtensible Hiper Text Markup Language*
XML	*eXtensible Markup Language*
XSD	*eXtensible Schema Definition*

SUMÁRIO

1 INTRODUÇÃO

Com o desenvolvimento tecnológico proporcionado nas últimas décadas (anos 70 até os dias atuais), muitas foram às inovações que hoje influenciam de forma direta o dia a dia das pessoas. Destacam-se dentre estas tecnologias aquelas voltadas para o setor de computação móvel. Conforme afirma Lemos (2007), "os dispositivos móveis são a ferramenta mais importante de convergência midiática hoje", fato este ligado a popularização do acesso à internet por meio destes dispositivos, bem como a constante agregação de funcionalidades antes presentes somente em computadores e notebooks, como: compartilhamento de fotos, acesso às redes sociais, leitura de notícias em tempo real, acesso a serviços de meteorologia, informações sobre o trânsito, entre tantas outras.

Na visão de McFaddin et al. (2008) a computação móvel prevê um mundo no qual humanos carregam dispositivos contendo todos os dados de sua vida cotidiana, e, usam estes dispositivos para se comunicar uns com os outros e desempenharem funções chaves, relacionadas a sua localização e contexto. Os autores comentam ainda que o propósito de comunicação está bem desenvolvido por meio de aplicações de voz, e-mail e mensagens. Contudo, no que diz respeito a localização e contexto dependem fundamentalmente da aplicação de serviços para prover informações.

Em termos conceituais de tecnologia, quando mencionado o termo "aplicação de serviços", está sendo feita uma referência direta a tecnologias de serviços web, por meio das quais os dispositivos móveis (assim como dispositivos de outras plataformas), comunicam-se de forma transparente.

Para Pilioura *et al.* (2005), o paradigma de serviços web é atualmente considerado a mais promissora e rápida evolução tecnológica para o desenvolvimento de aplicações em ambientes abertos, distribuídos e heterogêneos. A proliferação deste paradigma coincide com o significante aumento na capacidade de processamento dos dispositivos móveis, tanto em

quesitos de hardware quanto de software. A combinação dos dois mundos é considerada de suma importância para a indústria da computação nos próximos anos.

Quanto a conceituação de Serviços web Hamad, Saad e Abed (2010) atingem seu objetivo de forma neutra com relação a tecnologia, provendo interfaces bem definidas para funcionalidades distribuídas, as quais são independentes de plataforma de hardware, sistema operacional ou linguagem de programação.

Baseado nestas visões este trabalho se propõe a apresentar o desenvolvimento de um aplicativo baseando-se na tecnologia de serviços web, promovendo ainda um entendimento sobre o macro ambiente envolvido em uma solução desta natureza, voltada para o consumo de serviços por dispositivos móveis.

Para tanto, faz-se necessário uma breve conceituação realizada desde o histórico de desenvolvimento da Web, bem como o estudo dos dois principais protocolos e arquiteturas de Serviços Web, SOAP (*Simple Object Access Protocol*) e RESTful.

Segundo Tidwell, Snell e Kulchenko (2001), SOAP se coloca no topo da pilha de tecnologias para serviços web como protocolo padrão para empacotamento de mensagens compartilhadas por aplicações. Baseado em XML (*Extensible Marckup Language*) provê mecanismos para descrição e descoberta de Serviços (respectivamente WSDL - *Web Service Definition Language* e UDDI - *Universal Description, Discovery and Integration*). Trata-se de um meio robusto e seguro de desenvolvimento de aplicações orientadas a serviços, além de ser regulamentado pelo W3C[1] (*World Wide Web Consortium*).

Contrariando as afirmações do parágrafo anterior, o Protocolo SOAP não seria a solução ideal para o desenvolvimento uma aplicação capaz de promover o acesso ágil a informação, caso o cliente seja um dispositivo móvel, por este ser menos favorecido em questão de processamento, memória e acesso à internet. Segundo Hamad, Saad e Abed (2010) SOAP mostrou-se muito verboso, devido ao fato principalmente de sua base estar em padronização XML, enquanto que o paradigma RESTful, empregando a serialização de objetos utilizando JSON (*Javascript Object Notation*) mostrou-se mais eficiente nestes dispositivos.

[1] *The World Wide Web Consortion* – Comunidade internacional fundada por Tim Berners-lee, onde uma equipe trabalha em tempo integral em conjunto com organizações membro e o publico em geral para o desenvolvimento de padrões para a Web

Segundo Fonseca e Simões (2007), muito embora o padrão XML seja recomendado pela W3C para anotação de documentos, associado ainda as inegáveis vantagens trazidas por este padrão para o ambiente de representação e transferência de dados, proporcionada por sua robustez, flexibilidade e relativa facilidade, o XML possui limitações. Para embasar esta afirmação são apresentadas como desvantagens do modelo XML as seguintes características: sintaxe demasiadamente verbosa e redundante, dificuldade na criação de analisadores (*parsers*), reduzido conjunto de tipos de dados suportados pelos requisitos básicos de processamento, e sintaxe difícil de ser manipulada por humanos.

Para mitigar as desvantagens apresentadas pelo XML quanto aplicado à plataforma móvel, buscou-se uma alternativa que inicialmente se popularizou através da tecnologia Ajax (*Asynchronous JavaScript and XML*) para serialização de objetos, o JSON. Por meio deste padrão de serialização de objetos torna-se mais simples para o navegador (*browser*) obter uma estrutura de dados em formato JavaScript[2] a partir de uma *string* do que empregando XML. Muito embora JSON não esteja diretamente ligado ao JavaScript, devido a sua simplicidade de expressão e de entendimento tanto por humanos como por máquinas, este é um padrão que vem surgindo como alternativa ao emprego do XML.

1.1 PROBLEMATIZAÇÃO

A crescente necessidade de comunicação extrapolou o universo humano a tempos, atingindo o habitat de máquinas, dotadas de softwares, os quais detêm informações que nós humanos queremos compartilhar. O problema desta integração de dados de forma transparente reside nas diferentes plataformas de softwares, hardwares e linguagens de programação.

O estudo aqui apresentado baseou-se na necessidade de integração entre um software de ERP, rodando sob a plataforma *desktop*[3], e um software utilizado por equipes de vendas embarcado na plataforma móvel utilizando como dispositivo o iPad® da Apple®. Frente a estas duas plataformas computacionais distintas, dados referentes a pedidos de vendas, cadastros de produtos, clientes, entre outros necessitavam ser transportados de maneira transparentemente.

---

[2] Linguagem de programação interpretada, empregada em códigos executados em páginas no lado cliente, ou seja, interpretada pelo navegador do cliente.

[3] Sinônimo da plataforma de computadores pessoais.

Para possibilitar essa integração foi estudada a tecnologia de serviços web que conforme afirmação de Tidwell, Snell e Kulchenko (2001) são interfaces posicionadas entre o código da aplicação e o usuário deste código, mesmo que este usuário se trate de outra aplicação. Desta forma, aproveitando-se da interface padronizada provida por serviços web, aplicações podem se comunicarem de forma transparente e interoperável, transpondo barreiras impostas por quaisquer tipos de plataformas de hardware, sistemas operacionais e/ou linguagens de programação.

Como resultado do estudo sobre serviços web emergiu a necessidade de desenvolvimento de uma aplicação baseada nesta tecnologia para intercambiar dados de forma transparente entre as plataformas computacionais envolvidas.

E, baseado nas afirmações sobre SOAP e RESTful, bem como sobre XML e JSON, quando aplicadas a dispositivos de menor poder de processamento e uma qualidade inferior de acesso à internet, propôs-se o embasamento da solução a ser desenvolvida no paradigma RESTful empregando JSON como método de notação e serialização de objetos.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Este trabalho tem como objetivo geral a proposição de uma arquitetura de serviços web que possibilite a integração de dados entre aplicativos de diferentes plataformas computacionais, com foco nos requisitos de transparência, interoperabilidade, segurança e baixa suscetibilidade a erros.

1.2.2 Objetivos Específicos

Tendo em vista o objetivo principal, é necessário o cumprimento de alguns objetivos de cujo mais específico.

- Promover o entendimento dos conceitos de serviços web de um modo geral principalmente no que ser refere aos paradigmas de comunicação;
- Propor uma visão lógica e física da arquitetura de serviços;
- Desenvolver a arquitetura de serviços baseada no requisito anterior;
- Elaborar um estudo de caso baseado nos resultados dos estudos prévios das tecnologias e arquiteturas de serviços web a serem empregadas;

- Realizar uma discussão dos resultados visando demonstrar a viabilidade da solução para uso em um mercado corporativo.

1.3 METODOLOGIA

Para elaboração deste trabalho de conclusão de curso foi adotada a seguinte metodologia;

- Estudo e fundamentação do histórico evolutivo da Web até a concepção do conceito de serviços web, bem como a caracterização dos mesmos segundo revisão da literatura;
- Análise das soluções tecnológicas de comunicação disponíveis para desenvolvimento de soluções de serviços web;
- Proposição da arquitetura voltada à integração de dados de diferentes plataformas;
- Desenvolvimento da aplicação de serviços web proposta pelo levantamento de requisitos bem como a apresentação dos resultados obtidos;
- Apresentação dos resultados obtidos com a adoção da arquitetura proposta, enfatizando suas vantagens a solução anterior.

1.4 ESTRUTURA DO TRABALHO

Para uma melhor apresentação e consequente entendimento da temática proposta, este trabalho está dividido em quatro capítulos, onde o primeiro trata da introdução, problemática, objetivos e metodologia. O segundo capítulo tem por objetivo promover uma contextualização do assunto segundo conceitos históricos desde o surgimento da Web até a concepção do conceito de serviços web.

No capitulo três apresenta-se a arquitetura proposta, detalhando seus conceitos e componentes, promovendo um entendimento das técnicas e tecnologias usadas em conjunto formando a arquitetura final. O capítulo quatro promove uma visão do estudo de caso, onde a arquitetura proposta no capítulo três é aplicada sobre em um cenário, bem como uma análise sobre os resultados desta aplicação.

Por último, são apresentadas as considerações finais sobre o trabalho, bem como, são apresentados os trabalhos futuros.

2 SERVIÇOS WEB

A área de Tecnologias da Informação e Comunicação é extremamente dinâmica, inovações acontecem cotidianamente, novos softwares são escritos nas mais diversas linguagens, em diferentes plataformas e para os mais diversos fins. Esta diversidade de soluções promovem desafios e para viabilizar determinado negócio, algumas destas tecnologias distintas devem interagir para atingir um objetivo comum. É neste cenário que os serviços web têm um grande destaque.

Conforme Turtschi et al. (2002) serviços web foram criados para resolver os problemas de interoperabilidade entre aplicações, de diferentes sistemas operacionais, linguagens de programação e modelos de objetos. Zhao (2010) conceitua serviços web simplesmente como uma interface programável acessivel para outras aplicações atravéz da Web.

Atualmente, esta tecnologia vem sendo utilizada nos mais variados domínios, contudo, não foi a primeira tecnologia desenvolvida com este propósito. Anteriormente ao surgimento dos serviços web, outros padrões de comunicação já eram adotados, por exemplo, CORBA (*Common Object Request Broker Architecture*), inicialmente utilizado por sistemas UNIX e DCOM (*Distributed Component Object Model*) desenvolvido pela Microsoft.

O emprego de serviços web teve um grande impulso com o surgimento da WEB 2.0, fase na qual o conteúdo deixou de ser meramente estático para tornar-se dinâmico, dando início as aplicações web mais robustas, permitindo que os usuários passassem de expectadores a agentes ativos na construção de conteúdo. Porém, simplesmente gerar conteúdo não faz sentido se o mesmo não for compartilhado, e é justamente no compartilhamento de conteúdo entre aplicações web que o emprego de serviços web popularizou-se (FILHO, 2009).

Para entender melhor o contexto em que os serviços web se inserem será apresentada nas próximas seções uma breve revisão sobre sua evolução da web.

2.1 HISTÓRICO DA WEB

A maior invenção do homem no campo tecnológico depois do transistor, sem dúvida foi a Web. Partindo da ideia de organização de conteúdo por Vanevar Bush's em seu famoso artigo “As We May Think” de 1945, na qual propôs uma máquina batizada por ele de “Memex”, a qual por meio de um processo de código binário, foto células e fotografias instantâneas, permitiam fazer e seguir referências cruzadas em microfilmes (BERNERS-LEE, 1996).

A evolução da Web pode ser identificada em versões. Evans (2008), em uma apresentação expõe as versões iniciais atuais e futuras da web, pontuando características e tecnologias que marcaram e ainda marcarão a evolução da Web. De forma sucinta, serão apresentados alguns pontos considerados essenciais para um bom entendimento e separação de cada uma das fases, uma vez que uma em especial, tem muito em comum com o assunto abordado por este trabalho.

2.1.1 Web 1.0

Web 1.0 ou “A Web somente de leitura” (EVANS, 2008), inventada por Tim Berners-Lee em 1989, teve o primeiro *browser*[4] em outubro 1990 e o primeiro servidor web em novembro de 1990. Nesta versão os sites eram basicamente estáticos, com atualizações irregulares e continham basicamente elementos textuais, menus e ícones de navegação e imagens.

Quanto a tecnologia de motores de busca, estes eram caracterizadas por grandes índices, porém com técnicas de consulta rudimentares. Em 1993 foi lançado o *World Wide Web Worm* (WWWW) o primeiro site de busca, que somente considerava o título e cabeçalho das páginas web. Em 1994, Yahoo e WebCrawler foram os primeiros a procurarem conteúdo no texto completa das páginas, e em 1995 surgiu o Altavista que utilizava enormes índices e linguagem natural de busca com operadores booleanos. De um modo geral, nesta fase as tecnologias de busca se preocupavam basicamente em como minimizar o tamanho dos índices, enquanto a relevância era ignorada.

[4] Em Português “Navegador”, é um programa de computador que efetua requisições a um servidor web e retorna as páginas ao usuário.

2.1.2 Web 2.0

A versão 2.0 da Web é entendida como "A Web de leitura, gravação e execução". Uma das muitas definições para a Web 2.0 cunhada por Tim O'Reilly em 1999, potencializa todo o contexto inovador desta nova verão se comparada a versão 1.0. A dinâmica subjacente da Web 2.0 sugerido por Evans (2008) baseia-se em três grandes pilares:

Cauda longa – Tradicionalmente, lojas somente armazenavam as visitas, devido às limitações de espaço, ao contrário na Web 2.0, que sem restrições de espaço, podem armazenar qualquer coisa.

Dados Sociais – O sucesso da Web 2.0 agrega uma maior quantidade e qualidade dos dados, bem como, uma maior facilidade de navegação. O aproveitamento do lado social da Web, habilitando os usuários a criarem seus próprios dados (por exemplo, páginas indexadas do Yahoo, Google, Revisão de produtos da Amazon, informações da Wikipédia, etc.) e aplicações (provendo acesso a dados RSS, serviços web, etc.), e empregando seu comportamento como filtro (por exemplo, mecanismos de recomendação, algoritmos de ranqueamento, marcação, etc.) são outras características desta fase da Web (EVANS, 2008).

Efeito viral e a Sabedoria das multidões – Com surgimento das "Webs Apps"[5] na era da Web 2.0, quanto mais usuários acessam uma aplicação web, consequentemente maior será o volume de dados gerados, e mais valorizada essa aplicação será. O conceito de "Sabedoria das multidões" está fortemente ligado à ideia de que decisões tomadas por muitos na maioria dos casos é melhor que uma decisão tomada por um único indivíduo. Baseado neste argumento esta sabedoria é aproveitada em sistemas de votação, marcação, blogs, ranqueamento de páginas, entre outros. Na verdade este foi o fato que levou o Google™ a comprar o YouTube®[6] por perceber o número elevado de acessos que este site possuía.

Ainda segundo O´Reilly (2005), alguns pontos marcantes da era Web 2.0 podem ser mencionados, entre eles:

- Emprego de serviços e não softwares empacotados;
- Associação de versões beta ao conceito de agilidade;
- Fontes de dados difíceis de serem recriadas à medida que mais pessoas as utilizam tornando-as ricas em informação;

[5] Aplicações que apresentam funcionalidades semelhantes as conhecidas na plataforma desktop, porém hospedadas em um servidor Web e acessadas por meio de um Navegador.

[6] Site de compartilhamento de vídeos <www.youtube.com>.

- Confiança nos usuários como desenvolvedores paralelos;
- Aproveitamento da Inteligência Coletiva;
- Software além do nível de um simples dispositivo;
- Interfaces de usuários limpas propiciando uma experiência de uso mais rica aos usuários.

2.1.3 Web 3.0

A versão 3.0 da Web, que vem se desenvolvendo, muito embora esteja longe de alcançar seus objetivos se propõe a introduzir conceitos semânticos a Web, dando significado ao grande volume de informação nela contido, e possibilitando o entendimento destes significados não somente por humanos, como também por dispositivos computacionais. Na visão de Hendler (2009) pode-se ver a Web 3.0 a integração de tecnologias semânticas proporcionando aplicações Web de larga escala.

2.2 CONCEITO DE SERVIÇOS WEB

A abordagem de serviços utilizando a plataforma Web, ou meramente Serviços Web, consiste em uma tecnologia que emergiu com a Web 2.0. É importante ressaltar que esta tecnologia, assim como a Web, evolui constantemente. Exemplo disto, é a utilização do conceito de semântica na área, os chamados '*Semantic Web Services*', ou simplesmente Serviços Web Semânticos (FILHO, 2009).

Partindo para uma abordagem tipicamente SOAP de serviços web, conforme definido pelo W3C (2004) um serviço web é um sistema de software designado para suportar a interação de máquina-a-máquina de forma interoperável sobre uma rede. Tendo uma interface descrita em um formato processável por máquinas (especificamente WSDL). Outros sistemas interagem com o serviço web da maneira indicada por sua descrição usando mensagem SOAP, tipicamente empregando HTTP com serialização em XML em conjunto com outros padrões Web.

Segundo Tidwell, Snell e Kulchenko (2001), o termo Serviço Web pode ser definido como "uma funcionalidade de uma aplicação acessível por meio de uma interface de rede construída utilizando tecnologias padrões de Internet". Tal funcionalidade, justamente por ser baseada nos protocolos de internet, torna-se acessível para qualquer outra aplicação

independente da linguagem ou da plataforma para a qual foi desenvolvida, desde esta aplicação, esteja apta a se comunicar segundo um determinado protocolo.

De uma forma mais detalhada um Serviço Web pode ser entendido como uma interface de rede posicionada entre o código da aplicação e o usuário deste código, atuando como uma camada de abstração, possibilitando o acesso a qualquer aplicação que conheça esta interface.

Em outra abordagem apresentada por Gonsalves (2009), serviços web são definidos como “um tipo de lógica de negócio exposta por meio de uma interface de serviço para uma aplicação cliente” (Figura 1).

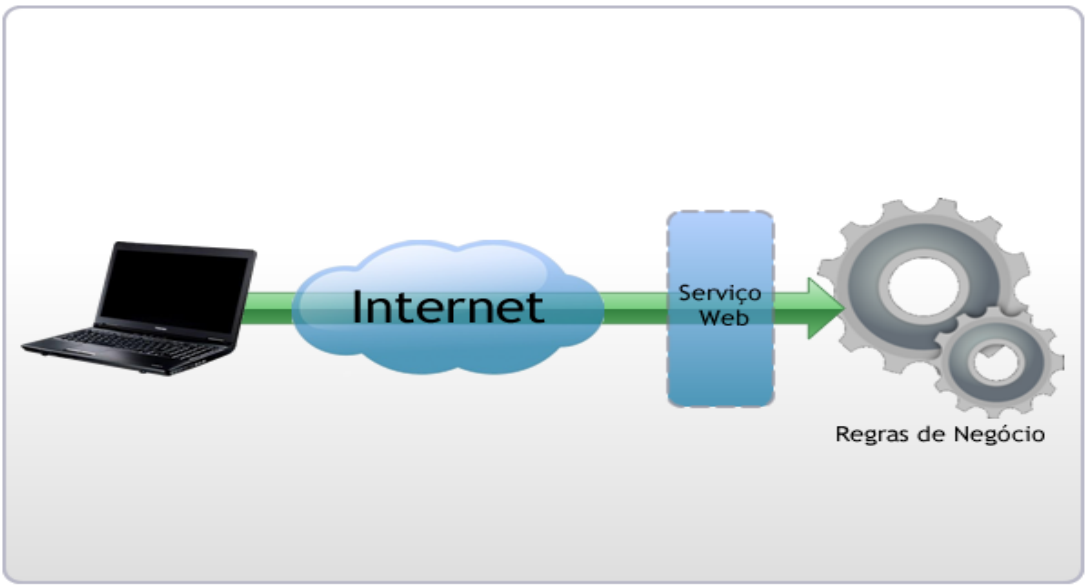

Figura 1 - Acesso a regras de negócio de uma aplicação com uso de serviços web
Fonte: Adaptada de Tidwell, Snell e Kulchenko (2001)

2.3 PROCOLO SOAP

Conforme visto anteriormente, SOAP está diretamente ligado ao padrão XML, e faz uso do mesmo para serialização de objetos e comunicação de dados entre o lado cliente e o lado servidor. “SOAP coloca-se na pilha de tecnologias para *web services* como protocolo padrão para empacotamento de mensagens compartilhadas por aplicações” (TIDWELL, SNELL e KULCHENKO, 2001).

2.3.1 Histórico

SOAP iniciou em 1998 e em 1999 a Microsoft™ lançou a versão 1.0 do protocolo, época que ainda não havia sido criado nenhuma linguagem de esquema ou tipos de sistema para XML. Portanto, nesta época o foco maior do protocolo era definir tipos de sistema, derivados de tipos primitivos formando tipos de dados semelhantes a estruturas e composições acessadas por posição como em vetores. Somente depois de ter esses tipos representacionais definidos é que eram modelados de fato os tipos comportamentais, como métodos e operações (BOX, 2001).

Em 2000 a IBM™ começou a trabalhar no SOAP 1.1, e em 2001 o W3C lançou a linguagem WSDL. Ainda em 2000, UDDI foi desenvolvido pela OASIS (*Organization for the Advancement of Strutured Information Standards*) permitindo a publicação e descoberta de serviços web. Com suporte de grandes empresas estavam sendo desenvolvidas as principais tecnologias envolvidas em um Serviço Web que emprega o protocolo SOAP (GONSALVES, 2009).

Ainda segundo Box (2001), podem ser claramente observadas duas fases no desenvolvimento do protocolo SOAP após o início do projeto. A primeira de 1999 a 2000, em que alguns fatos marcantes ocorreram, dentre eles:

- O protocolo foi finalmente batizado com nome oficial de “SOAP”;
- Avanço na linguagem de esquema do W3C, que levou o grupo de trabalho do SOAP a ver que a integração com esta linguagem deveria ser levada em conta tanto quanto o possível;
- Tentativa frustrada de inclusão de um tipo de metadados simples (CDL);
- Esforço para inclusão de suporte completo aos esquemas XML o que resultou no desenvolvimento de um compilador XSD (*XML Schema Definition*) em C++[7] utilizado internamente pelo grupo de trabalho do projeto.

A partir de 2001 o SOAP entra em uma nova fase, na qual evolui muito, em grande parte devido a concomitante evolução do XML e da linguagem de esquemas. A seguir são citadas algumas características do protocolo SOAP:

- A especificação de esquema XML (XML *Schema*) está estável e passa a ser proposta como recomendação;

[7] Linguagem de programação derivada de C, onde aparecem os conceitos de Orientação a Objetos.

- A fundação de um grupo de trabalho no W3C para o protocolo XML;
- A proximidade da definição de um metadados para SOAP (WSDL);
- Aceitação do protocolo SOAP pelos fabricantes de software que vem suportando SOAP gradativamente.

2.3.1.1 Conceito de Serviços Web segundo SOAP

O princípio básico de funcionamento do SOAP é simples, consiste na descoberta do serviço registrado em um registro (UDDI) e o acesso ao mesmo (Figura 2).

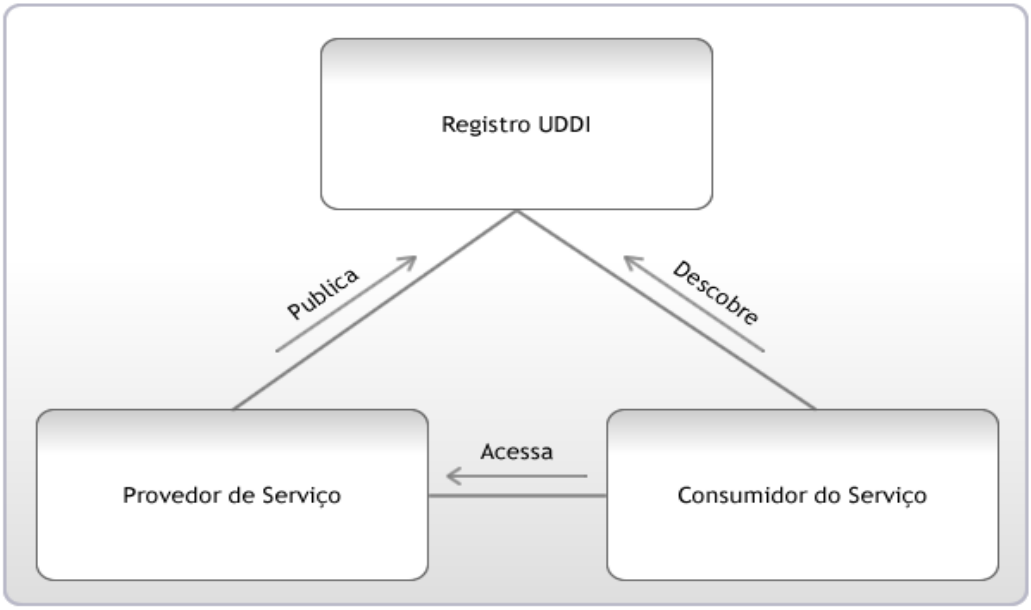

Figura 2 - A descoberta de um serviço mediante consulta a um registro UDDI
Fonte: Adaptada de Shomoyita e Ralph (2011)

No processo geral de publicação, descoberta e consumo de um Serviço Web padrão SOAP, algumas tecnologias estão envolvidas (GONSALVES, 2009). São elas: UDDI, WSDL, SOAP, HTTP (*Hyper Text Transfer Protocol*) e XML.

2.3.1.1.1 UDDI

A intenção de utilização do UDDI é permitir que aplicações que precisam comunicar-se umas com as outras por meio da Web, possam encontrar informações que viabilizem esta comunicação.

A especificação de UDDI para suporte a serviços web, consiste da implementação de bases de dados de serviços web *on-line* e distribuídas. Mais especificamente os registros

UDDI suportam o gerenciamento de meta-informação, as quais descrevem serviços em particular. Normalmente estas meta-informações são representadas em linguagem WSDL. Estes registros possibilitam ainda a descoberta de forma automática ou semiautomática da composição de serviços, podendo desta forma ser visto como um diretório para serviços web (BLAKE, SILVA, *et al.*, 2007).

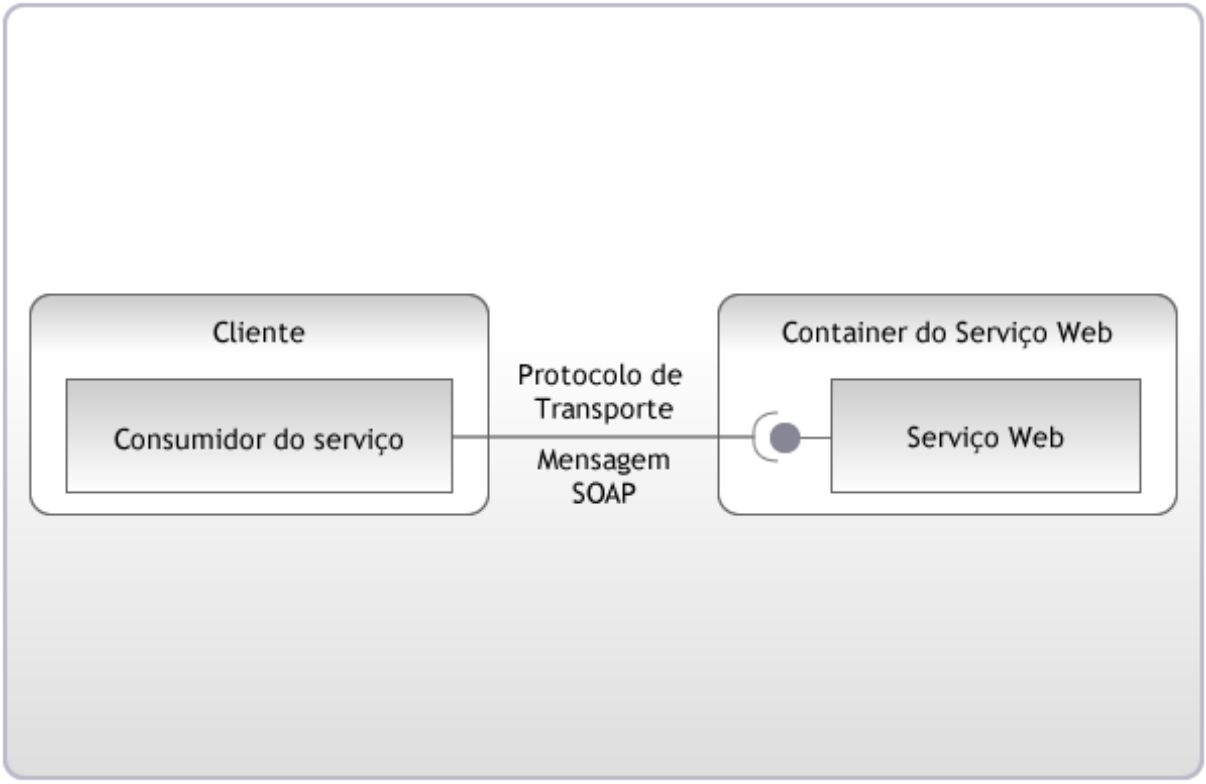

Figura 3 - Interface WSDL entre o Cliente e o Serviço Web
Fonte: Adaptada de Gonsalves (2009)

2.3.1.1.2 WSDL

Na arquitetura do protocolo SOAP o registro UDDI deve apontar para um arquivo WSDL público, disponível na internet para potenciais consumidores do serviço web. O arquivo WSDL é escrito em linguagem de definição de *interface* IDL (*Interface Definitivo Language*), e define a interface do serviço web, informando os tipos de mensagens suportadas, porta, protocolo de comunicação, operações suportadas, localização, e demais informações que o desenvolvedor do serviço web julgar necessária informar para quem deseje consumir o serviço. Para assegurar a interoperabilidade, um serviço web padrão é preciso que o consumidor e o produtor compartilhem e entendam as mensagens. Sendo assim, este é o foco do WSDL (ORT, 2005).

2.3.1.1.3 SOAP

SOAP é um protocolo de aplicação padrão para serviços web e está relacionada diretamente à tecnologia XML (GONSALVES, 2009), ou seja, SOAP é uma aplicação da especificação XML (TIDWELL, SNELL e KULCHENKO, 2001). Isto significa que SOAP é basicamente uma implementação de envelopes baseado em XML para transporte de informações, bem como um conjunto de regras para traduzir tipos de aplicações e plataformas específicas para representação XML.

Um envelope SOAP lembra muito a marcação da estrutura básica de uma página HTML (*Hyper Text Markup Language*), contendo um cabeçalho (opcional) e um corpo. No cabeçalho do envelope, são informados dados referentes a configurações de entrega da mensagem, autenticação ou regras de autorização ou contexto de transações. No corpo por sua vez, o conteúdo da mensagem a ser transmitida é informado.

Figura 4 - Estrutura de um envelope SOAP

Fonte: Adaptada de Tidwell, Snell e Kulchenko (2001)

2.3.1.1.4 Protocolo de Transporte

Comumente utilizado, o protocolo mais convencional da Internet é o HTTP, embora para configurações que exijam maior segurança no tráfego das informações o uso de HTTPS

(HTTP seguro) seja encorajado. Não tanto comum quanto o HTTP, qualquer outro protocolo de transporte como o TCP/IP (*Transfer Control Protocol/Internet Protocol*), SMTP (*Simple Mail Transfer Protocol*) ou FTP (*File Transfer Protocol*), podem ser empregados para transferência de mensagens (GONSALVES, 2009).

2.3.1.1.5 XML

O padrão XML hoje é largamente utilizado, por facilitar a compreensão por humanos e até certo ponto por máquinas, do conteúdo que descrevem. Este padrão de marcação teve seu início em 1998 e evoluiu juntamente com o SOAP. Porém, o mesmo não está tão fortemente atrelado ao SOAP, quanto o SOAP a ele, visto que a base de funcionamento do protocolo SOAP é fundamentada na troca de mensagens serializadas e envelopadas em documentos XML. Outros padrões como o RESTful (que será abordado mais adiante) também podem empregar o XML como meio de serialização de objetos.

Em conjunto com os esquemas permite a definição de tipos e validação de dados, uma vez que o mesmo não é somente utilizado no envio de mensagens, como também no arquivo WSDL.

Devido ao fato do XML não estar vinculado a nenhuma aplicação, plataforma, Sistema Operacional, ou linguagem de programação específica a interoperabilidade do serviço web pode ser garantida. Por exemplo, uma aplicação escrita em Perl executando em Windows, pode enviar uma mensagem a uma aplicação Java executando em um Sistema Operacional Unix (TIDWELL, SNELL e KULCHENKO, 2001).

```
<SOAP:Envelope xmlns:SOAP="http://schemas.xmlsoap.org/soap/envelope/">
    <SOAP:Header>
        <!— conteúdo do cabeçalho do Envelope SOAP —>
    </SOAP:Header>
    <SOAP:Body>
        <!— conteúdo do Corpo do Envelope SOAP —>
    </SOAP:Body>
</SOAP:Envelope>
```

Figura 5 - Estrutura de um Envelope SOAP
Fonte: Adaptada de Curbera et al. (2002)

2.4 ARQUITETURA REST

REST (*Representational State Transfer*) é um estilo arquitetural para sistemas multimídia distribuídos que enfatiza a generalização das interfaces, a escalabilidade da

integração entre os componentes e a instalação independente dos mesmos (FIELDING, 2000). De forma direta REST é um paradigma, uma arquitetura que reúne um grupo de critérios a serem incorporados ao projeto de aplicações distribuídas, apesar de não existir qualquer conexão direta com algum protocolo especificamente. A definição de RESTful ainda é citada por outros autores como, Kamaleldin e Duminda (2012) como sendo baseada em recursos, os quais são identificados por URIs únicas. Estes recursos são acessados e manipulados usando um conjunto de métodos uniformes (GET, POST, PUT e DELETE), onde cada recurso pode ter uma ou mais representação (XML, JSON, Text, etc), as quais são transferidas entre o cliente e o serviço, durante a invocação ao mesmo.

2.4.1 Histórico

A arquitetura REST foi idealizada e definida por Roy Thomas Fielding em sua tese de PhD, muito embora não tenha sido inicialmente definido com o propósito para o qual é utilizado hoje. De fato o propósito inicial foi "designar uma arquitetura de design e desenvolvimento para a Web moderna..." (FIELDING, 2000). Porém, partindo desta apresentação, a arquitetura REST difundiu-se como uma forma simplificada para implementação de serviços web ou "RESTful *Web Services*" (RICHARDSON e RUBY, 2007).

O fato de interligar REST e HTTP beneficiou a arquitetura REST de duas formas, sendo: (a) pelo fato do HTTP ser o protocolo normalizado pelo W3C como padrão para transmissão de mensagens na Web; e (b) todos os sistemas interligados na Web o implementam, ou seja, o grupo de sistemas compatíveis com REST é abrangente. Além disso, por utilizar a própria Web como infraestrutura de distribuição e acesso, torna-se inteiramente portável, não restando qualquer dependência adicional de *hardware* ou *software* (FILHO, 2009) e (KAMALELDIN e DUMINDA, 2012).

Como resultado da interligação entre REST e HTTP, Richardson e Ruby (2007), definiram uma arquitetura orientada a recursos chamada ROA (*Resource Oriented Architecture*), em que esta seguia fielmente a estrutura REST ao mesmo tempo em que empregava o HTTP.

Algumas empresas emergentes da Web 2.0 começaram então a migrar suas API's do protocolo SOAP para o emergente paradigma RESTful, algumas delas citadas a seguir:

- Google – Tendo marcado sua API SOAP (SOAP *Search* API) como obsoleta desde o ano de 2006, em 31 de agosto de 2009 finalmente declarou a descontinuidade definitiva (GOOGLE INC., 2009). A partir desta data oferece exclusivamente uma API (*Application Programming Interface*) inteiramente baseada em classes JavaScript em que os serviços web são disponibilizados por meio de RESTful. Muito embora esta API tenha representado um avanço, a mesma está marcada como obsoleta desde novembro de 2010 e pode sair do ar brevemente para dar lugar a atual "*Custom Search* API", a qual elimina as classes JavaScript e disponibiliza toda a API somente por meio de serviços web RESTful;
- Twitter[8] – Disponibiliza uma API REST para desenvolvimento de aplicações integradas fazendo uso do protocolo de autenticação. O Auth[9], o qual permite acesso aos dados do usuário sem a necessidade de informar as credenciais do mesmo (tipicamente um par contendo usuário e senha), por meio do redirecionamento de agentes de usuário (HAMMER-LAHAV, 2010).

2.4.2 Arquitetura ROA

ROA é composta por alguns conceitos (Endereçabilidade, Estado Não-Persistente, Conectividade), como não possuir um estado persistente e considerar importante a descrição dos serviços, utilizando como pontes para comunicação (HONG, 2012). Funciona basicamente como qualquer outro serviço web, que recebe uma requisição detalhando uma ação a ser executada e retorna uma resposta detalhando o resultado obtido. Oberva-se que para ROA (ou RESTful), tanto a forma como a ação será executada como seu escopo de execução é discriminado na requisição (FILHO, 2009). Isso se deve ao fato de que em RESTful o estado das requisições não é armazenado, e cada requisição é única.

A seguir serão detalhados os cinco componentes principais da arquitetura ROA: Recurso, Representação, Identificador Uniforme, Interface Unificada e Escopo de Execução.

[8] Microblog muito conhecido na web, por permitir postagens de até 140 caracteres.

[9] Padrão aberto para autorização de usuários

2.4.2.1 Recurso

Trata-se de uma abstração ou conceito relevante no domínio tratado pelo serviço em questão, ficando a cargo do projetista a seleção de qualquer objeto do domínio, seja ele real ou fictício, concreto ou abstrato. Exemplos de recursos:

- Universidade Federal de Santa Catarina – Campus Araranguá;
- A Banda AC/DC®;
- A intenção de compra dos clientes que acessam um e-commerce[10];
- A localização geográfica da praia Morro dos Conventos;
- A coleção de DVD's da série 24 Horas®.

Conforme pode ser observado pelos exemplos citados qualquer item que possa ser definido em um objeto sendo considerado e tratado por REST como um recurso, até mesmo uma coleção de objetos como o exemplo da coleção de DVD's, pode ser considerada um recurso (FILHO, 2009).

2.4.2.2 Representação

Na arquitetura REST, os serviços manipulam as representações dos recursos, uma vez que estes são abstratos e não podem ser constituídos fisicamente. Por exemplo, não seria possível trafegar um carro, ou uma pessoa pela rede, mais é possível trafegar sua representação. Desta forma fica claro que conceitualmente uma representação é qualquer conjunto de dados útil sobre o estado de um recurso (RICHARDSON e RUBY, 2007).

Tecnicamente técnica pode-se dizer que uma representação consiste na serialização de um recurso, empregando-se para isso uma sintaxe específica. Dentre as sintaxes mais conhecidas e utilizadas destacam-se o XML, XHTML (*Extensible Hypertext Markup Language*), JSON e RDF (*Resourse Description Framework*). Cada sintaxe de serialização é definida pelo cabeçalho da requisição, conforme ilustra a Tabela 1. Assim como RDF que é uma linguagem que pretende padronizar o uso de XML para definição de recursos, existem outras que empregam o XML para anotação de dados (FILHO, 2009).

[10] Termo utilizado para designar comércio eletrônico de produtos na Web.

Formato	Cabeçalho
XML	application/xml
XHTML	application/xhtml+xml
JSON	application/json
RDF	application/rdf+xml

Tabela 1 - Formatos para representação de valores e respectivos cabeçalhos HTTP

Diante da diversidade de opções para serialização de recursos, um serviço web pode, eventualmente, prover acesso empregando diferentes métodos para serializar seus recursos, promovendo assim uma maior interoperabilidade entre sua interface e seus potenciais consumidores. Nestes casos, a requisição deve ser objetiva quanto a notação do que se espera como retorno. Para tanto é utilizado um cabeçalho HTTP chamado *Accept*. O campo *Accept* do cabeçalho de requisição HTTP pode ser utilizado para especificar certos tipos de mídias aceitáveis como resposta a requisição (FIELDING, *et al.*, 1999). A Figura x ilustra a utilização deste campo do cabeçalho HTTP.

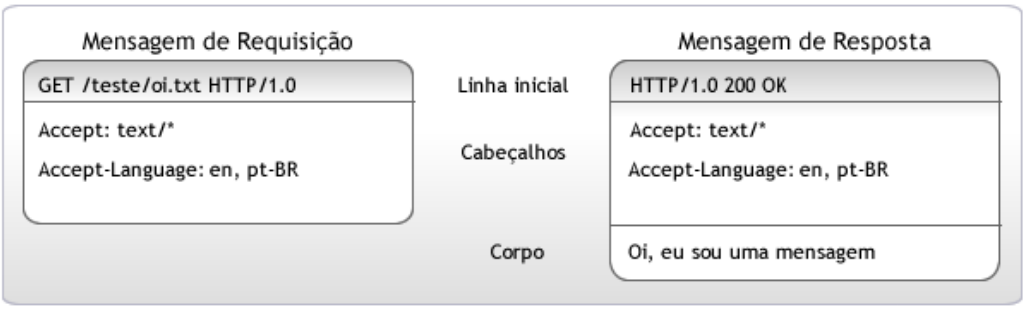

Figura 6 - Exemplo de requisição e resposta HTTP
Fonte: Adaptada de Gourley e Totty (2002)

2.4.2.3 Identificador Uniforme (URI)

Na arquitetura REST, um recurso necessariamente é referenciado por pelo menos um identificador uniforme ou URI (*Universal Resource Identifier*), que possui as funções de identificação e localização deste recurso. Desta fora caso um objeto não seja referenciado por nenhum URI, este não pode ser considerado um recurso. Por outro lado, um recurso pode ser referenciado por um número ilimitado de URIs (O'REILLY, 2007).

O termo URI é comumente confundido com o termo URL (*Universal Resource Locator*), porém o URL é na verdade, um subconjunto do endereçamento URI, ou seja, o URI

é um esquema de endereçamento mais abrangente e único para cada recurso disponível na Web, seja ele um documento HTML, uma imagem ou um serviço web. Um URI é composto por três partes, conforme ilustra a figura abaixo:

Figura 7- Exemplo de um URI

De acordo com o exemplo pode-se afirmar que: o documento "home.html" pode ser acessado através do protocolo HTTP, no domínio www.nce.ufrj.br sob o diretório "/cursos/htmlbasico/".

Desta forma as três partes que compõe um URI são:

- Protocolo de acesso (http://);
- Domínio (endereço onde o recurso encontra-se hospedado);
- O caminho para o recurso propriamente dito (diretório + recurso).

Outra forma de apresentação de URIs pode ser como "http://www.domain.com/application/resource/", neste caso o recurso "resource" é apontado, estando este na aplicação "application" hospedada no domínio "www.domain.com" e acessada pelo protocolo HTTP. Nesta forma de apresentação diz-se que o URI é descritivo. Alguns exemplos de URIs:

- http://www.acme.com/restfulws/users
- http://www.acme.com/restfulws/users/100
- http://www.example.com/bookstore/book/123
- http://www.weather.com/weather/2008?location=Paris,France
- http://www.movies.com/catalog/titles/movies/123456
- http://www.movies.com/categories/aventure

2.4.2.4 Interface Unificada

O paradigma RESTful define que toda ação a ser executada é definida diretamente pelo protocolo HTTP. O protocolo HTTP define cinco métodos principais: GET, HEAD,

POST, PUT e DELETE. Todos estes métodos segundo RESTful são aplicáveis aos recursos, ou objetos gerenciados pelo serviço.

O HTTP ainda oferece os métodos OPTIONS, TRACE e CONNECT, embora até o momento estes não tenham sido absorvidos pela arquitetura ROA. O uso destes métodos do protocolo HTTP agrega simplicidade na exposição de métodos em formato de serviços web, uma vez que podem ser acessados por um URI padrão.

Uma vez que o consumidor do serviço conheça os recursos oferecidos, automaticamente conhece os processos de criação, alteração, exclusão e recuperação destes recursos, bastando para isso alterar o método do protocolo na chamada do serviço. Esta característica promove uma maior interoperabilidade e conceitua-se como interface unificada (FILHO, 2009).

O paradigma RESTful como mencionado anteriormente emprega os métodos do HTTP para manipulação de recursos, desta forma, cada método do protocolo HTTP corresponde a uma ação a ser tomada sobre um recurso, desta forma torna-se muito simples a criação e entendimento de casos CRUD (*Create, Read, Update, Delete*) para a manipulação de recursos. A Tabela 2 relaciona os métodos HTTP, com a ação, sua equivalência em SQL (*Structured Query Language*) e descreve a funcionalidade de cada item (RICHARDSON e RUBY, 2007).

Método HTTP	Ação	Instrução SQL	Operação
POST	**C**reate	INSERT	Cria um novo recurso, inserindo dados
GET	**R**ead	SELECT	Obtém os dados de um recurso
PUT	**U**pdate	UPDATE	Atualiza os dados de um recurso
DELETE	**D**elete	DELETE	Exclui um recurso e seus dados

Tabela 2 - Utilização dos métodos HTTP e sua equivalência com SQL

Segundo Gonsalves (2009), outros métodos do protocolo HTTP não frequentemente usados são:

- HEAD é idêntico ao método GET, exceto que este não transfere o recurso. Por este motivo, é útil para verificar a viabilidade do *link*, ou obter o tamanho do recurso;
- TRACE ecoa de volta para o consumidor do serviço a requisição recebida;

- OPTIONS retorna ao consumidor do serviço os dados referentes as opções de comunicação disponíveis para uma requisição/resposta a um recurso especificado por um URI, permitindo assim ao cliente determinar quais as opções e/ou requerimentos associados a um recurso, ou as capacidades do servidor, sem que haja necessidade de obter o recurso em si;
- CONNECT é utilizado em conjunto com um *proxy* que pode dinamicamente ser configurado para iniciar um túnel (uma técnica pela qual o HTTP atua como um empacotador para vários protocolos de rede).

2.4.2.5 Escopo de Execução

Quanto ao escopo de execução a abordagem do paradigma RESTful defende o emprego do URI contendo não somente o endereço do recurso, mais também qualquer parâmetro necessário a identificação única e/ou outros dados para manipulação do recurso afetado.

A Figura 8 exemplifica uma requisição para o endereço fictício galeria.com, onde supostamente uma galeria de imagens encontra-se acessível e conta com a abordagem RESTful na implementação de sua API de acesso.

```
01 GET /foto/123 HTTP/1.1
02 Host: galeria.com
```

Figura 8 - Requisição RESTful utilizando o método GET do protocolo HTTP

Neste caso, o recurso foto está sendo acessado pelo método GET e o servidor entende que deve retornar o recurso (foto) identificado pelo código 123 passado como parte do URI.

No exemplo de requisição apresentado abaixo (Figura 9) é efetuado o acesso ao mesmo URI, somente o método do protocolo HTTP para acesso foi alterado, de GET para DELETE em relação ao exemplo anterior. Esta alteração basta para o servidor entender que o recurso identificado pelo código 123 deve ser excluído.

```
01 DELETE /foto/123 HTTP/1.1
02 Host: galeria.com
```

Figura 9 - Requisição RESTful utilizando o método DELETE do HTTP

2.4.3 Princípios adicionais do Paradigma RESTful

Em adição as características comentadas anteriormente, um serviço que se diz RESTful, pode e deve apresentar mais alguns princípios. São eles, Endereçabilidade (*Addressability*), Estado Não-Persistente (*Statelessness*) e Conectividade (*Connectivity*). Princípios estes que serão abordados em maiores detalhes a seguir.

2.4.3.1 Endereçabilidade

Segundo Laurent et al. (2001), diferentemente do paradigma RPC (*Remote Procedure Call*), onde um único URI serve de referência para endereçamento do serviço como um todo, e cada método é representado por um parâmetro a ser passado, não fazendo parte do endereço em si, no paradigma RESTful, cada porção de dados pode ser endereçada, isto é, receber um URI específico para sua referenciação. Segundo Richardson e Ruby (2007), serviços são classificados como endereçáveis quando seu conjunto de dados é exposto como um conjunto de recursos, cada um com seu respectivo URI. As Figura 10 e Figura 11 ilustram essa diferenciação no endereçamento de recursos.

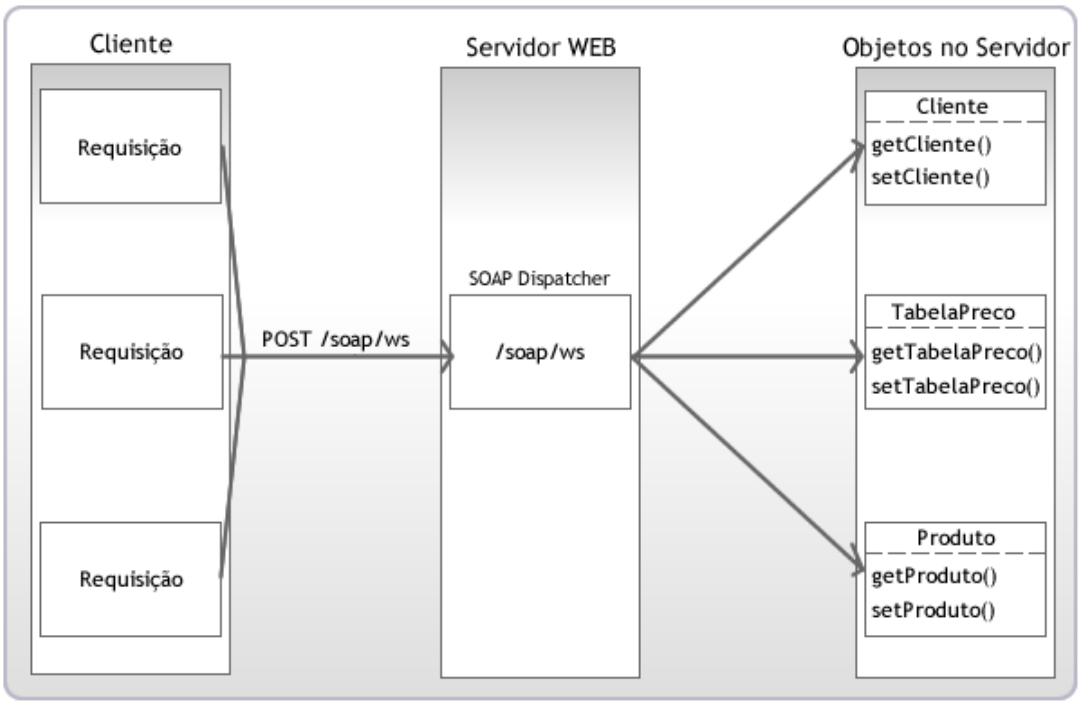

Figura 10 - Endereçamento no padrão SOAP (XML-RPC)

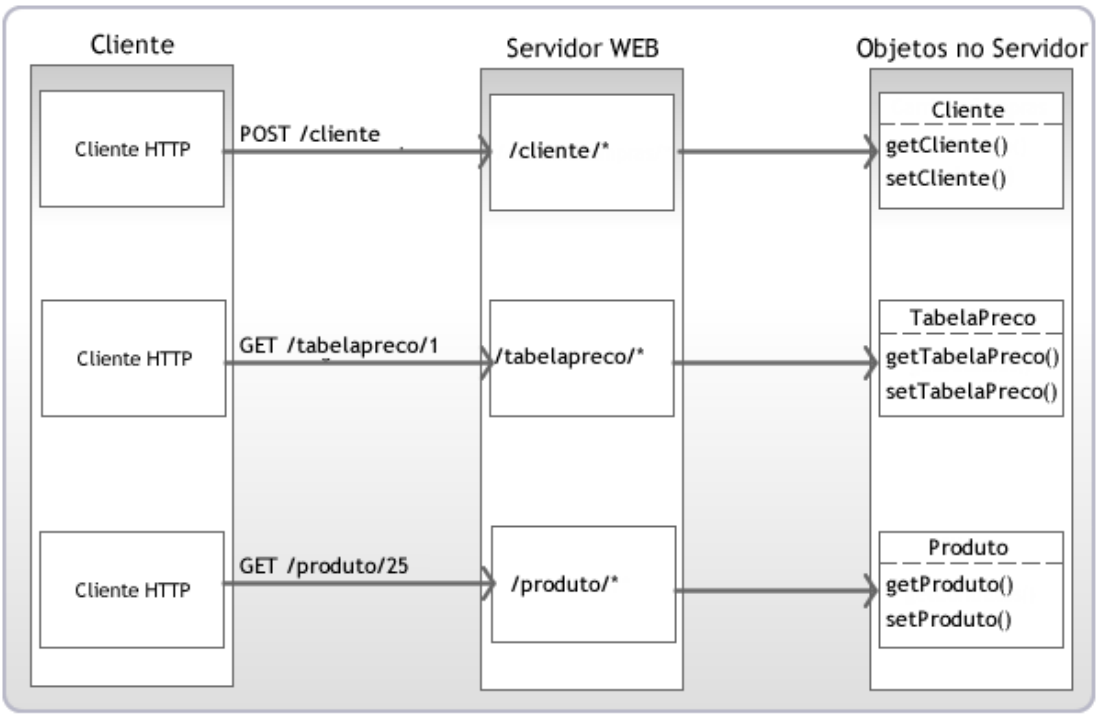

Figura 11 - Endereçamento de recursos segundo o paradigma RESTful

2.4.3.2 Estado Não-Persistente

Estado Não-Persistente (*Stateless*) está relacionado à propriedade do paradigma RESTful na qual cada requisição a um recurso é única. Para Richardson e Ruby (2007), o conceito de não persistência em uma requisição HTTP está relacionado ao fato de possibilitar o total isolamento da conexão. De forma sucinta, em cada conexão todas as informações necessárias para o servidor executar a requisição devem ser enviadas pelo cliente, e o servidor não armazena informações da sessão.

De forma mais prática, o paradigma RESTful dita que todo pedaço de informação importante, deve ser apresentado como um recurso, tendo um URI de acesso próprio. A técnica de Estado Não-Persistente diz por sua vez que, cada estado deve também ser endereçado por seu próprio URI. O cliente não deve ter que induzir o servidor para um determinado estado, para torná-lo receptivo a determinada requisição.

Na Web em certos momentos o usuário se vê frustrado por não estar apto a utilizar o botão "voltar" de seu navegador. Isto ocorre quando o usuário executou uma ação irrevogável, como a compra de um produto em um *e-commerce*, ou postou um texto em um *blog*, por exemplo. Este comportamento de Estado Persistente (*Stateful*) viola o princípio de Estado Não-Persistente.

As Figuras 12 e 13 diferenciam os acessos a serviços segundo os padrões de Estado Não-Persistente e Estado Persistente, ilustrando as requisições e respostas obtidas no acesso a um motor de busca.

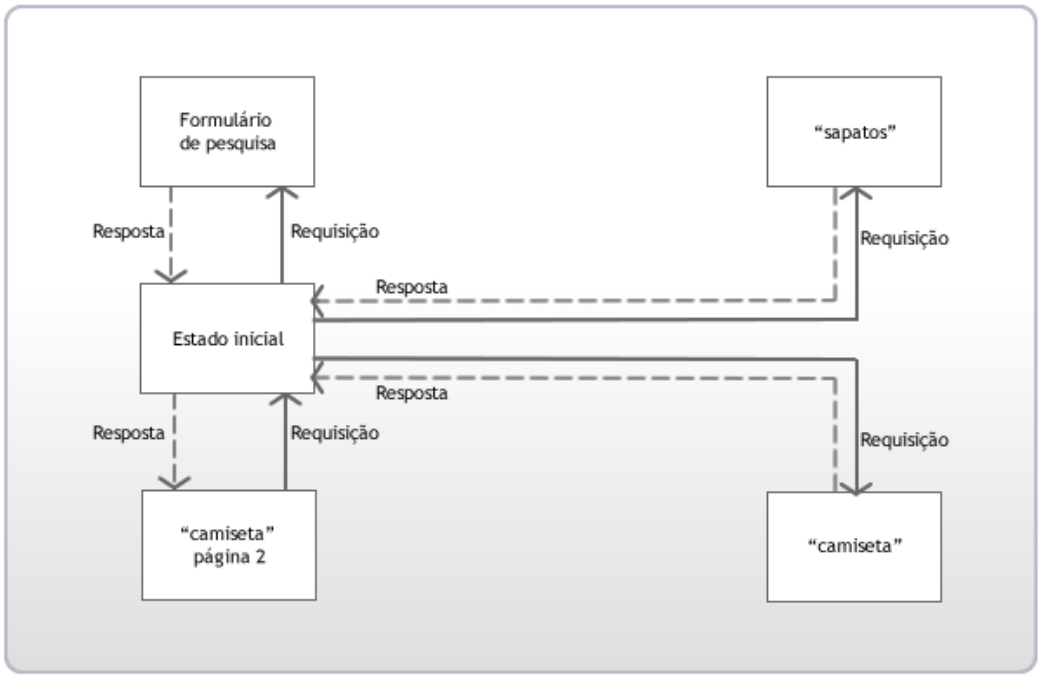

Figura 12 - Motor de busca *Stateless* (Estado Não-Persistente) em q eu cada requisição gera uma resposta e a conexão é perdida

Fonte: Adaptada de Richardson e Ruby (2007)

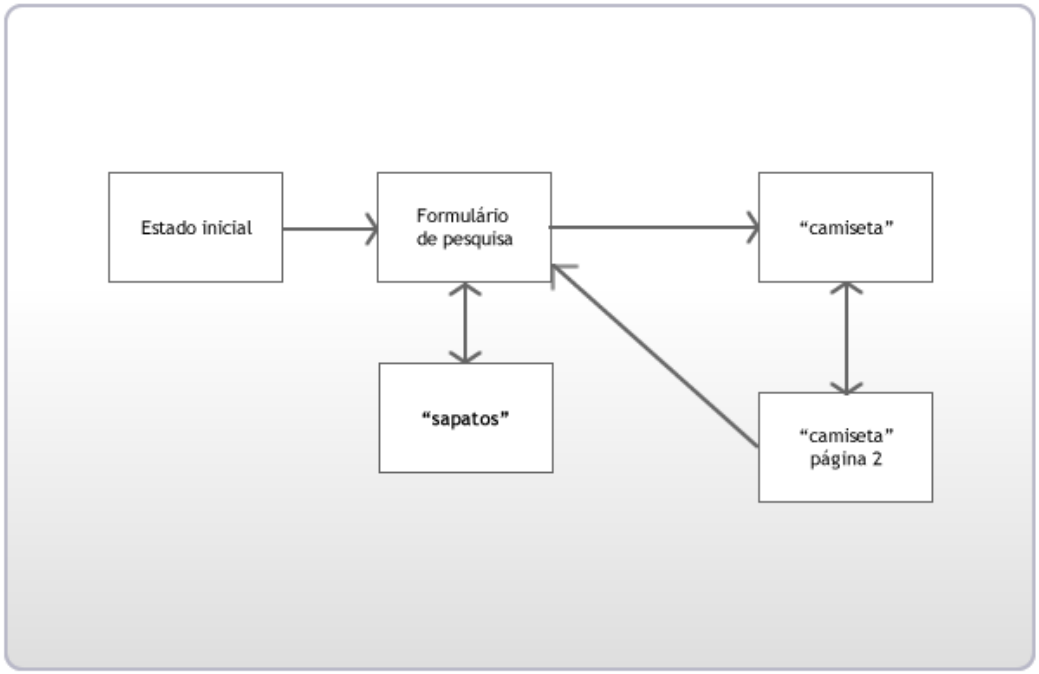

Figura 13 - Motor de Busca *Stateful* (Estado Persistente) em que cada requisição inicia uma sessão bidirecional, e a conexão não se encerra quando o cliente obtém a resposta do servidor

Fonte: Adaptada de Richardson e Ruby (2007)

2.4.3.3 Conectividade

Segundo Richardson e Ruby (2007), “um recurso deve apontar para outros em sua representação”. Gonsalves (2009) faz uma analogia a teoria de grafos, onde enfatiza que nesta teoria, um grafo é dito conectado se todos os pares de vértices distintos deste puderem ser conectados através de algum caminho. Assim como fortemente conectado se contiver um caminho direto de *u* para *v* e um caminho direto de *v* para *u*, para cada par de vértices *u*, *v*. REST por sua vez prega que os recursos devem ser o mais conectado possível. Deste modo, formula que conectividade pode ser associada à RESTful como: “Hipermídia com um motor de estado de aplicação”.

Em outras palavras o Princípio da Conectividade em ROA está ligado a representações conectadas a outras representações, ou seja, os recursos gerenciados pelo serviço apresentam-se conectados entre si.

É correto ainda fazer uma analogia do Princípio da Conectividade em ROA com a utilização de *hiperlinks* para realizar as ligações dos documentos de hipermídia disponibilizados na Web, em que os usuários navegam entre diferentes páginas apenas seguindo *links* que apontam de uma página à outra.

2.4.4 Descrição de Serviços

Ao contrário do padrão SOAP que tem a linguagem WSDL bem definida como meio de descrição de serviços, em RESTful ainda não há um padrão bem formado, no entanto uma abordagem conhecida por WADL (*Web Application Description Language*) vem demonstrando ser promissora para ocupar o lugar de instrumento padrão de descrição de serviços RESTful.

2.4.4.1 WADL (*Web Application Description Language*)

Ao formular o padrão WADL, Hadley (2009), membro integrante do grupo de pesquisas da Sum Microsystems, elencou alguns pontos positivos de sua utilização, entre eles:

- Suporte ao desenvolvimento de ferramentas para modelagem de recursos;
- Geração automática de código para a manipulação de recursos;
- Configuração de cliente e servidor, a partir de um único formato portável.

Apesar da extensão do tema, por se tratar de um ambiente fechado e controlado, onde não é desejável nem aceitável o consumo dos serviços por terceiros, pesquisas sobre este assunto não foram aprofundadas.

2.5 APLICAÇÕES DE SERVIÇOS WEB

Nesta sessão serão apresentadas algumas soluções tecnológicas que envolvem o uso de Serviços Web, nas mais diferentes situações. O propósito aqui é demonstrar a versatilidade desta tecnologia, e os benefícios que a mesma proporciona, utilizando para isso exemplos documentados de seu uso.

Machado et al. (2008) apresenta uma solução modular baseada em software livre, objetivando a troca de informações entre médico e paciente, objetivando minimizar a necessidade de contato físico entre ambos, permitindo dentre outras a liberação de leitos hospitalares.

Um forte apelo da solução supracitada é a possibilidade de integrar diversas plataformas de desenvolvimento sem custo (relacionado a *softwares*). Para tornar possível esta solução, foram empregados entre outros componentes, dispositivos móveis e serviços web, a linguagem de programação Java®, e o servidor de aplicação GlassFish®.

Em via geral, o sistema baseia-se nos dados coletados por um medidor cardíaco, enviados por meio de um serviço web para o servidor de aplicação, e persistido em uma base de dados. Neste ambiente os dados coletados do paciente, tornam-se disponíveis para o médico, por meio de dispositivos móveis, os quais consomem as informações via os serviços web.

O emprego de serviços web seguindo a arquitetura RESTful foi proposto por McFaddin et al. (2008), para a coleta de dados de dispositivos de usuários de um determinado espaço de comércio, como *Shopping*, Estações Rodoviárias e de Metro, Hospitais, entre outros. De posse destes dados os usuários do espaço de comércio recebem informações úteis sobre as oportunidades de negócio proporcionadas pelo local onde o usuário do dispositivo móvel está situado.

Este mesmo conceito de comercio eletrônico empregando serviços web, é citado por Zhao (2010), porém com foco direcionado a técnicas para promover a confiabilidade nos serviços.

Uma proposta mais ousada de emprego de serviços web, mais especificamente sob o modelo arquitetural RESTful é apresentada por Ratinimittum e Piromsopa (2012), onde é apresentado um protótipo de sistema de arquivos[11] em rede utilizando RESTful. Na proposta é introduzido o RFS (*RESTful File System*), com intenção de permitir á uma máquina utilizar o espaço em disco de diversos servidores, como uma unidade de rede, formando um sistema de arquivos altamente escalável. Basicamente os dados são armazenados em alguns servidores nós, empregando alguns scripts PHP[12] (Hypertext Preprocessor). Os clientes deste sistema de arquivos utilizam a API RESTful para contatar cada nó, os quais suportam somente dois métodos GET e POST.Cada arquivo é dividido em blocos, promovendo assim um melhor aproveitamento do espaço em disco de cada nó, bem como o balanceamento de uso da rede. Por fim, para um maior desempenho ainda é empregado um sistema de cache baseado no algorítimo de reposição LRU (*Least Recently Used*).

Quanto a segurança e dispositivos móveis Bertram e Kleiner (2012) salientam que os recursos nativos para consumo de serviços SOAP atualmente oferecidos pela plataforma Android™[13] não são suficientes para atender a demanda de aplicativos corporativos. Em seguida apresentam uma solução própria com recursos de segurança sobre o protocolo SOAP, a qual baseia-se principalmente na codificação, por meio de extesões do protocolo, mantendo assim a flexibilidade do SOAP de um lado, e a viabilidade de programação do outro.

Baseando-se no atual aumento na capacidade de processamento, capacidade de armazenamento, capacidade de memória, e até mesmo das redes móveis, Kamaleldin e Duminda (2012), apresentam um *framework* leve para hospedagem de serviços web em dispositivos móveis, fazendo ainda uma comparação entre as arquiteturas SOAP e RESTful. Como resultado deste trabalho concluiram que a solução RESTful consome menos recursos e é mais eficiente para a implementaçao de serviços web hospedados em dispositivos móveis.

Outro emprego da tecnologia de serviços web é apresentado por Arroqui et al. (2012), onde é descrito um aplicativo executando sob a plataforma Android®, que acessa serviços web para controlar um simulador de uma fazenda, com o propósito de ajudar na tomada de decisão baseando-se em dados metereológicos, de sensores remotos georeferenciados, e em outros bancos de dados públicos.

---

[11] Termo que identifica estruturas de dados especificadas para gerenciar dados dispostos em múltiplos arquivos, armazenados de forma contínua, tipicamente empregado em discos rígidos e outros dispositivos de armazenamento de dados.

[12] Linguagem de programação interpretada largamente utilizada em aplicações Web.

[13] Sistema operacional de código aberto desenvolvido sob uma versão do Kernel do Linux pela Open Handset Alliance, um grupo formado por empresas de tecnologia e telecomunicações como Google, Motorola, Samsung, entre outras.

Por fim, cita-se o trabalho de Chao, Nengcheng e Liping (2012) tendo como objetivo designar e implementar um *link* persistente e flexivel entre sensores instalados em satélites estacionários. A solução visa coordenar, organizar e agregar serviços web aos sensores para atender a exigências de um cenário complexo de observação da Terra.

3 ARQUITETURA PROPOSTA

Visando promover uma solução de sincronização de dados de forma transparente entre plataformas computacionais distintas, mais especificamente entre plataformas móveis, neste caso o iOS® da Apple®, e um sistema de ERP (*Enterprise Resource Planning*) executando em computadores convencionais, foram levantados requisitos de forma a compreender as necessidades a serem atendidas pela solução. A abordagem do sistemas e dispositivos móveis não está no escopo deste trabalho.

De modo a facilitar o entendimento, bem como abstrair o problema a ser solucionada, a arquitetura aqui proposta encontra-se dividida em uma visão lógica e uma visão física.

3.1 VISÃO LÓGICA

A Figura 14 apresenta a visão lógica da solução proposta por este demonstrando de uma forma mais superficial os componentes que fazem parte da arquitetura proposta, bem como as ligações entre os mesmos.

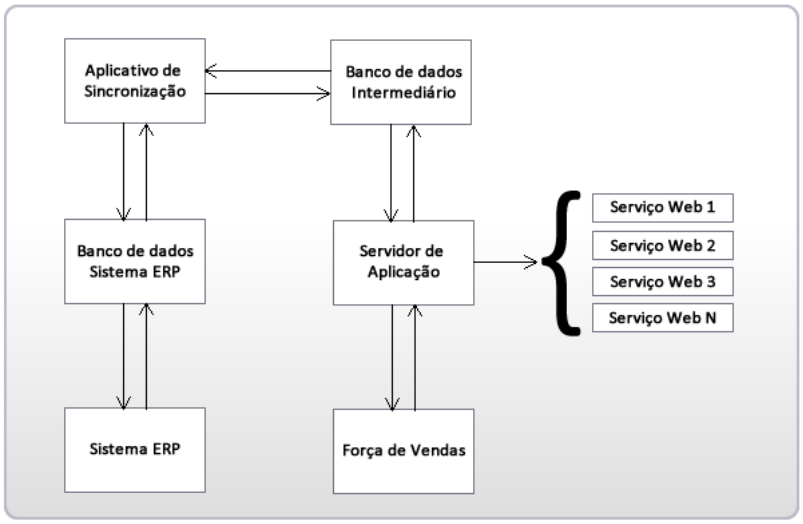

Figura 14 - Visão lógica da solução de serviços web proposta

A visão lógica observada na Figura 14 é decomposta dividindo-se em 6 componentes sendo:

- Sistema ERP – Sistema de gerenciamento do negócio da empresa.
- Banco de dados do sistema ERP – Local de persistência dos dados gerados e consumidos pelo sistema ERP.
- Aplicativo de sincronização – Software responsável pela captação das alterações realizadas tanto no lado dos sistemas móveis quanto no sistema ERP.
- Base de dados intermediária – Utilizada pelo Serviço web, também denominada "banco online". Recebe os dados enviados pelo serviço web, e vindos do banco de dados do sistema ERP através do aplicativo de sincronização.
- Servidor de Aplicação – Software onde estão hospedados os serviços web a serem consumidos pela força de vendas.
- Força de vendas – Software móvel destinado a equipe de vendas em campo, com a finalidade de efetuar pedidos, abrir cadastros de clientes, entre outras funcionalidades.

3.2 VISÃO FÍSICA

A visão física apresenta os componentes tecnológicos detalhando como estes se interconectam e contribuem para o desenvolvimento da solução final.

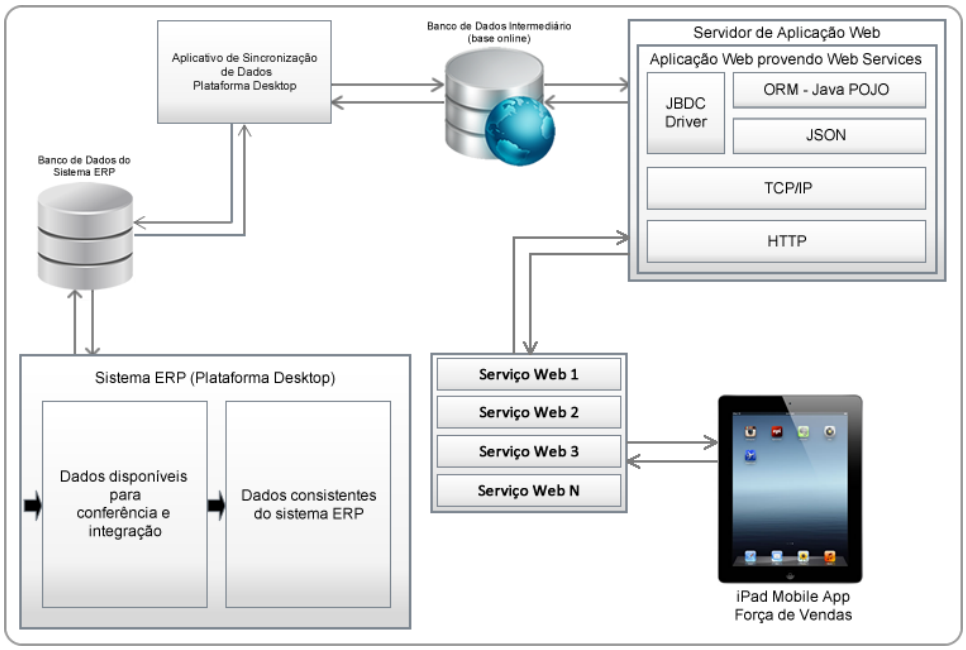

Figura 15 - Visão física da solução proposta

Para um melhor entendimento da visão física são apresentados os componentes tecnológicos, sendo:

Sistema ERP: Este componente é um software desenvolvido para executar em computadores pessoais, cujos dados são persistidos em uma base de dados executando em um servidor. Este sistema deve prover dados que serão utilizados pelo sistema móvel utilizado pela equipe de vendas, bem como ser alimentado por informações provenientes do mesmo sistema móvel.

Bando de dados do sistema ERP: Sistema Gerenciador de Base de Dados (SGDB), no qual os dados utilizados pelo sistema ERP são persistidos.

Aplicativo de Sincronização: Basicamente trata-se de outro software que roda na mesma plataforma do sistema ERP, tendo por finalidade comparar os dados da base de dados do sistema ERP com os dados persistidos na "base de dados online". A partir das divergências encontradas entre estas duas instâncias de base de dados, este aplicativo se encarrega de gerar scripts em linguagem SQL os quais mais tarde serão utilizados para atualizar os dispositivos móveis.

Outra funcionalidade para este software é atuar no processo inverso, ou seja, obter os dados dos dispositivos móveis, e com base nas diferenças encontradas nos dados persistidos na base de dados do sistema ERP, gerar scripts em linguagem SQL para refletir essas diferenças nesta base.

Banco de Dados Intermediário (Base online): Trata-se de uma base de dados utilizada para comparação com os dados persistidos na base de dados, alimentada pelo sistema ERP. Além deste propósito, esta mesma base de dados é utilizada para persistir os scripts a serem obtidos pelos dispositivos móveis.

Servidor de Aplicação: Constitui em peça fundamental na arquitetura sendo que no servidor de aplicação é publicado o sistema de serviços web, que será consumido pelos dispositivos móveis para troca de dados com a base online.

Este servidor de aplicação em sua arquitetura básica pode ser descrito como um conjunto envolvendo um driver de acesso a base de dados, uma camada desenvolvida para abstração de dados persistidos na base de dados e transformação dos mesmos em objetos, uma camada responsável por traduzir estes objetos computacionais em objetos serializados, utilizando a notação JSON, bem como uma camada de acesso ao protocolo TCP e implementação do protocolo de HTTP.

Com base nestes componentes o servidor de aplicação, juntamente com o aplicativo de serviços web devidamente publicado, é capaz de comunicar-se com o banco de dados e transpor os dados desta camada para uma interface web utilizando o protocolo HTTP, em notação JSON.

Serviços web: São as entidades do banco de dados convertidas em recursos RESTful. Atendendo aos requisitos de implementação do RESTful, todo recurso deve ser endereçado, desta forma, clientes, pedidos, produtos, e outras entidades do banco de dados são endereçadas e acessadas diretamente por meio de uma URL pela aplicação móvel. Para cada recurso diferentes ações podem ser desempenhadas alternando apenas o método HTTP utilizado.

Aplicação Móvel: Esta pode ser interpretada como a parte cliente desta solução, fazendo uma analogia a arquitetura cliente/servidor. Trata-se de um aplicativo desenvolvido especialmente para executar na arquitetura móvel da Apple®, no dispositivo iPad®[14], o qual comunica-se com o restantes dos componentes desta arquitetura pelos serviços web.

Para compor a arquitetura aqui proposta, ainda foram levados em consideração outros pontos, como interoperabilidade, desempenho, tolerância a falhas e questões de segurança.

3.2.1 Interoperabilidade

Interoperabilidade é a capacidade de diferentes sistemas de comunicar e compartilhar dados entre si. Ou seja, o serviço web a ser desenvolvido deve ser capaz de comunicar-se com qualquer dispositivo móvel, mesmo que o projeto inicial envolva apenas o tablete da Apple® (iPad®), em um futuro deve ser capaz de prover acesso aos mesmos dados para uma aplicação executando na plataforma Android® por exemplo, sem necessitar nenhuma alteração em seu código fonte.

Segundo o princípio da interoperabilidade e os padrões de serviços web podem perfeitamente, mesmo que não seja o caso deste projeto, permitir a comunicação tanto com dispositivos móveis, quanto sistemas desktops ou mesmo um sistema web de um site, sem que se faça necessária nenhuma alteração em sua implementação. Para garantia de interoperabilidade então é necessário a criação de uma interface de acesso única.

[14] Dispositivo móvel (Tablet) comercializado pela Apple®

3.2.2 Desempenho

Na solução oferecida anteriormente empregando Windows Mobile® e acesso direto a bancos de dados sobre a conexão do dispositivo móvel, o desempenho obtido não era satisfatório, devido ao fato de que os pacotes de dados trafegados serem maiores por levarem informações pertinentes a comunicação com o banco de dados, trafegarem sobre uma rede de acesso móvel de baixa qualidade, e em caso de queda não ter um controle de onde a operação foi interrompida, o que permitiriam uma continuação quando a comunicação fosse restabelecida.

Seguindo o princípio de mudança de paradigma, pela adoção de uma tecnologia de ponta em dispositivos móveis, o mecanismo de intercâmbio de dados não poderia ser de inferior capacidade, por este motivo a questão de desempenho é fator determinante na escolha das tecnologias envolvidas para o desenvolvimento da solução.

3.2.3 Tolerância a Falhas

Para evitar problemas encontrados na solução anteriormente adotada, algumas medidas referentes à tolerância a falhas devem ser adotadas. Dentre as medidas relacionadas para este projeto destacam-se a possibilidade de continuar um processo de sincronização, partindo do ponto em qual houve a queda de comunicação, minimizando assim o tempo despendido no processo, bem como a evitar uma verificação de consistência para identificar quais dados foram ou não foram transferidos.

Pensando em casos de implementações mais críticas, onde um maior número de dispositivos dependam da solução de serviços web, o servidor de aplicação deve possibilitar uma implementação em HA (*High Availability* – Alta Disponibilidade), ou seja, neste ambiente mais de um servidor de aplicação é empregado, sendo que um servidor mestre recebe e processa as requisições. Caso este servidor fique fora de produção por qualquer motivo, o segundo servidor deve imediatamente entrar em operação, suprindo assim a demanda de acesso dos clientes sem que haja uma interrupção no fornecimento dos serviços.

3.2.4 Segurança

Neste projeto, onde os dados trafegados entre sistema ERP e dispositivos móveis são críticos para a organização, deve ser garantida a integridade de acesso a estes dados.

O controle de acesso deve permitir que somente representantes com dispositivos autorizados e devidamente liberados pela empresa que utilizar essa solução de software deve estar apto a ter acesso aos dados da empresa. Para tal garantia algumas medidas de segurança se fazem necessárias, como a identificação e permissão de acesso mediante a verificação de um código de acesso e uma senha, desta forma somente conhecendo esses dados será possível consumir os serviços.

Em se tratando de tráfego de dados sob a plataforma web, deve ser também levado em conta a garantia de que mesmo que uma conexão autorizada esteja em curso, e que, terceiros interceptem dados desta conexão, não sejam capazes de utilizá-los. Para atender este requisito o servidor web deverá suportar a implementação do protocolo HTTPS, ou seja, HTTP empregando encriptação de dados com chaves assimétricas SSL (*Secure Socket Layer*).

4 ESTUDO DE CASO

No cenário atual da empresa, na qual foi desenvolvido este trabalho, é comercializado um software de força de vendas desenvolvido para a plataforma Windows Mobile, com o propósito de automatizar a força de venda das empresas. Este software possibilita cadastros e alterações cadastrais de clientes, emissão de pedidos e controle de visitas.

A sincronização dos dados entre aplicativo móvel e sistema ERP, consiste na recuperação de uma base de dados intermediária (conhecida no domínio da empresa por “base online”). Contendo registros com comandos SQL embutidos, responsáveis por inserir, apagar e/ou alterar dados armazenados no dispositivo móvel, referentes a dados cadastrais de clientes, produtos, tabelas de preços, regras de vendas entre outros.

A geração dos scripts de atualização do dispositivo móvel é tarefa de outra aplicação executando no desktop, que basicamente compara as duas bases de dados, do sistema ERP e a base online, gerando scripts com comandos SQL nativos (INSERT’s, UPDATE’s e DELETE’s), de forma a normalizar as duas bases de dados para um mesmo estado de consistência.

Outra tarefa desta aplicação além de comparação de dados é o processamento das operações necessárias para a equalização dos registros nos SGBDs (Sistemas de Gerenciamento de Bancos de Dados) do sistema ERP e dispositivo móvel. Uma vez no sistema ERP os dados são analisados pela empresa e então geram pedidos, cadastros de novos clientes e/ou alterações cadastrais nos clientes já existentes.

4.1 CONTEXTUALIZAÇÃO DO CENÁRIO

No cenário atual, é utilizado um software de plataforma móvel executando em sistemas Windows Mobile. O aplicativo móvel faz uso dos dados gerados pelo sistema ERP e atualiza o mesmo com dados inseridos pela equipe de vendas.

O processo de atualização de dados entre as plataformas móvel executando Windows Mobile® e *Desktop* acontece por meio de conexão direta ao SGBD executando no servidor das empresas, fazendo uso dos próprios serviços de transação disponibilizados por este SGBD.

O acesso a estes serviços pelo dispositivo móvel acontece por meio de conexão GPRS, 3G, EDGE, ou por meio de um computador, utilizando-se a conexão presente com a internet. Este processo de acesso direto a base de dados, principalmente via conexão de rede móvel, no Brasil ainda hoje é muito precário, muitas áreas não tem uma cobertura adequada, ocorrem muitas quedas de conexão, gerando instabilidade e lentidão no acesso.

As dificuldades tecnológicas citadas no parágrafo anterior enfatizam a dificuldade na realização da tarefa de sincronização entre aplicativo móvel e sistema ERP.

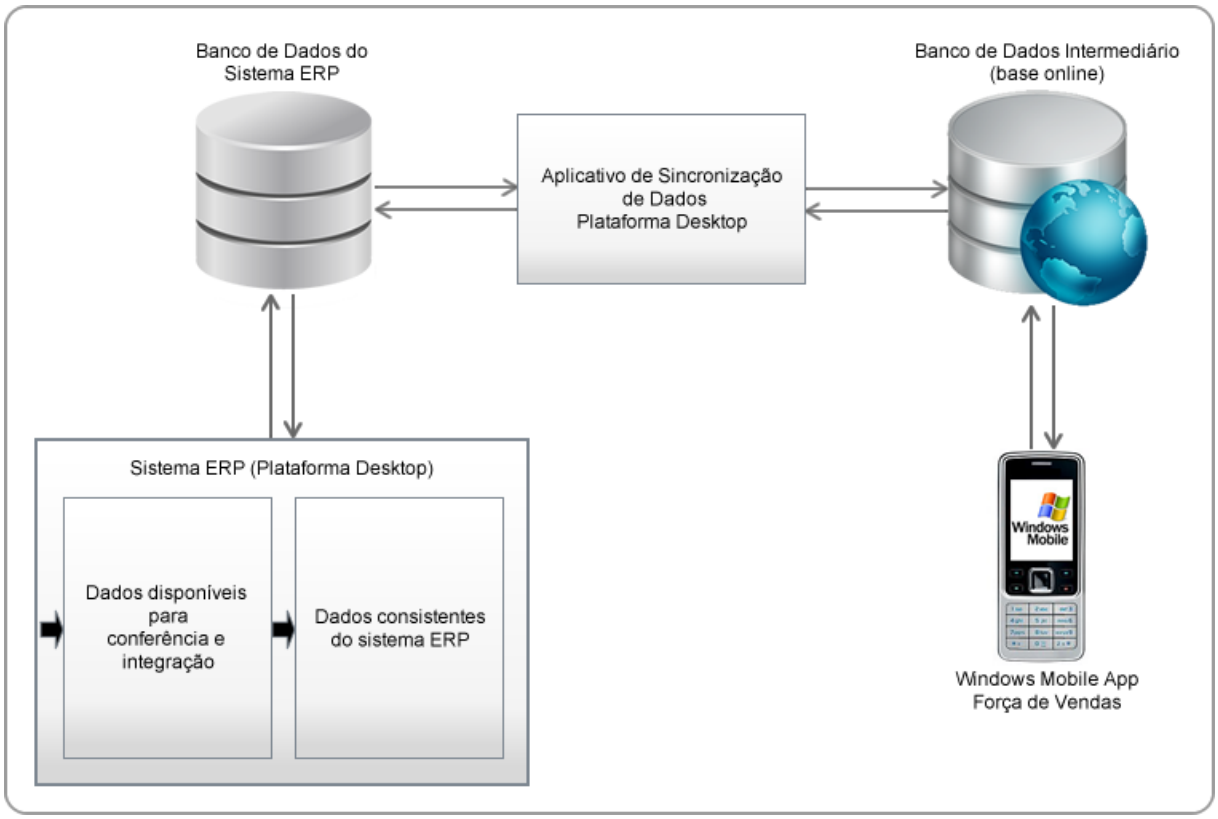

Figura 16 - Fluxo de intercâmbio de dados utilizado com a plataforma Windows Mobile (Cenário atual, utilizado antes da adoção da arquitetura proposta)

Para aplicar uma nova arquitetura neste cenário foram feitas análises dos seus principais pontos negativos, visando um melhor tratamento do problema, a sincronização de dados entre plataformas distintas. Baseando-se nesta análise, foram elicitados requisitos para o desenvolvimento da arquitetura proposta, destacando as necessidades a serem atendidas pela solução de forma a proporcionar o entendimento da mesma.

4.1.1 Requisitos

Requisitos são o foco dos estudos da Engenharia de Requisitos, a qual consiste no conjunto de tarefas que levam a um entendimento de qual será o impacto do software sobre o negócio, do que o cliente quer e de como os usuários finais irão interagir com o software (PRESSMAN, 2006).

Requisitos classificam-se como, Requisitos de Usuários expressos por meio de declarações em linguagem natural e de alto nível de abstração e Requisitos de Sistema, que abrangem as funcionalidades do sistema, seus serviços e restrições operacionais com um maior nível de detalhamento. Em outra abordagem Requisitos são classificados como Funcionais Não-Funcionais e de Domínio (SOMMERVILLE, 2007).

4.1.1.1 Requisitos Funcionais

Descrevem as funcionalidades do sistema, o que o sistema deve estar apto a fazer. Estes Requisitos dependem do tipo de software que será desenvolvido, das expectativas dos futuros usuários deste software bem como da maneira como estes foram coletados na organização. Uma vez expressados como requisitos de usuários, usualmente são descritos de uma forma bastante abstrata, no entanto estes requisitos descrevem o sistema de forma detalhada, documentando as entradas, saídas, exceções, etc. (SOMMERVILLE, 2007).

4.1.1.2 Requisitos Não Funcionais

Como o próprio nome sugere os Requisitos Não-Funcionais não visam funcionalidades específicas do sistema, sua preocupação intrínseca está relacionada ao desempenho, I/O[15] de dispositivos, representação de dados usadas nas interfaces do sistema, segurança, disponibilidade e outras propriedades emergenciais, o que implica em sua maior importância quando comparados aos Requisitos Funcionais (SOMMERVILLE, 2007).

[15] Termo em língua inglesa para representação de *Input*/*Output*, ou Entrada/Saída

Os Requisitos Não-Funcionais prendem-se somente ao sistema a ser desenvolvido, em alguns casos podem formar o processo que deverá ser adotado no desenvolvimento do sistema, incluindo especificações de padrões de qualidade, de design do produto e de processo a ser seguido. Surgem das necessidades dos usuários, relacionadas a restrições orçamentárias, políticas organizacionais, necessidades de interoperabilidade entre sistemas de software ou hardware heterogêneos, ou ainda por meio de fator externos relacionados a regulamentações de segurança ou legislação (SOMMERVILLE, 2007). Com base nestes conceitos os Requisitos Não-Funcionais podem ser classificados como:

- Requisitos de Produto – Especificam o comportamento do produto/software;
- Requisitos Organizacionais – Derivados das políticas e procedimentos das organizações consumidoras e de desenvolvimento;
- Requisitos Externos – Todos os requisitos derivados de fatores externos ao sistema e seu processo de desenvolvimento.

4.1.2 Levantamento de Requisitos para o Projeto

Com base nos conceitos de requisitos, e nas análises das expectativas a serem atendidas com a solução final, foram levantados os seguintes requisitos junto aos *stakeholders*:

- Promover uma interface padrão, disponível via rede e acessível na Web, capaz de fornecer acesso a dados persistidos em uma base de dados;
- Receber nesta mesma interface estruturas de dados e persisti-las na base de dados;
- Possibilitar o acesso a qualquer plataforma desde que se apresente em conformidade com os requisitos de autenticação;
- Autenticar e efetuar acessos à interface de comunicação de dados;
- Promover a paginação de resultados, para alavancar o desempenho em objetos de dados com muitos dados associados.

Com base neste levantamento prévio, após a realização de uma análise mais detalhada foram elicitados os seguintes Requisitos Funcionais e Não-Funcionais:

4.1.2.1 Requisitos Funcionais

Para um melhor efeito de documentação dos Requisitos Funcionais foi adotado como padrão de nomenclatura o código RQF_ seguido de uma numeração sequencial (Tabela 3). Juntamente com o código de cada requisito foi documentada uma descrição técnica abreviada, bem como uma classificação de prioridade (OB = Obrigatório; DE = Desejável).

Requisito	Descrição	Prioridade
RQF_0001	WS para acesso a dados de Clientes	OB
RQF_0002	WS para acesso a dados de Produtos	OB
RQF_0003	WS para acesso a dados de Grupos de Produtos	OB
RQF_0004	WS para acesso a dados de Tabelas de Preços	OB
RQF_0005	WS para acesso a dados de Representantes	OB
RQF_0006	WS para acesso/envio de Pedidos de Vendas	OB
RQF_0007	WS para acesso/envio de Itens de Pedidos de Vendas	OB
RQF_0008	WS para acesso ao Status do Serviço	DE
RQF_0009	Apresentar os dados de forma paginada para diminuir o tráfego	DE

Tabela 3 - Requisitos Funcionais

4.1.2.2 Requisitos Não-Funcionais

Ainda com base no levantamento prévio de Requisitos efetuado em conjunto com os *stakeholders*, também foram elicitados requisitos não funcionais a serem contemplados neste estudo de caso. Assim como para os requisitos funcionais, foi elaborado um código de identificação, que obedece a nomenclatura RQNF_ seguido por um número sequencial (Tabela 4).

Requisito	Descrição	Prioridade
RNF_0001	Validar o acesso mediante id + login + hash a todos os WS	OB
RNF_0002	Tratar todas as exceções e gravá-las em Log	DE
RNF_0003	Logar acessos autenticados realizados a qualquer WS	OB
RNF_0004	Utilizar o padrão de serialização JSON, possibilitando acesso a qualquer plataforma que o implemente e para a qual o acesso seja corretamente autenticado	OB
RNF_0005	Consultar e Persistir os dados em uma base de dados SQL	OB
RNF_0006	Tornar o acesso a todas as interfaces de WS visíveis na WEB	OB
RNF_0007	Retornar uma estrutura padrão para todos os serviços acessados contendo: • Código de retorno • Mensagem retornada • Página de dados atual • Número total de páginas • Estrutura de dados obtida para o WS	OB
RNF_0008	Utilizar um servidor de aplicação gratuito e robusto o bastante para prover os serviços	DE
RNF_0009	Utilizar o paradigma de programação MVC, visando a separação dos dados e regras de negócio, promovendo um acesso mais facilitado aos dados bem como uma maior manutenabilidade do projeto	DE

Tabela 4 - Requisitos Não-Funcionais

4.2 MODELAGEM DO PROJETO EM DIAGRAMAS UML

Para amparar o desenvolvimento do projeto com base nos requisitos levantados, foram elaborados alguns diagramas UML (*Unified Modeling Language*). Muito embora a definição da UML 2.0 especifique 13 diagramas diferentes que podem ser empregados na modelagem de um projeto, entendeu-se que para a especificação básica deste caso de uso eram necessários somente os diagramas de Caso de Uso, Diagrama de Classes, Diagramas de Pacotes e Diagrama de Sequência.

4.2.1 Diagramas de Caso de Uso

Devido ao fato de que a Aplicação Web que irá prover os serviços web se tratar de uma interface única, todo o projeto segundo a visão do usuário foi definido em um único diagrama de Caso de Uso, conforme apresentado pela Figura 17.

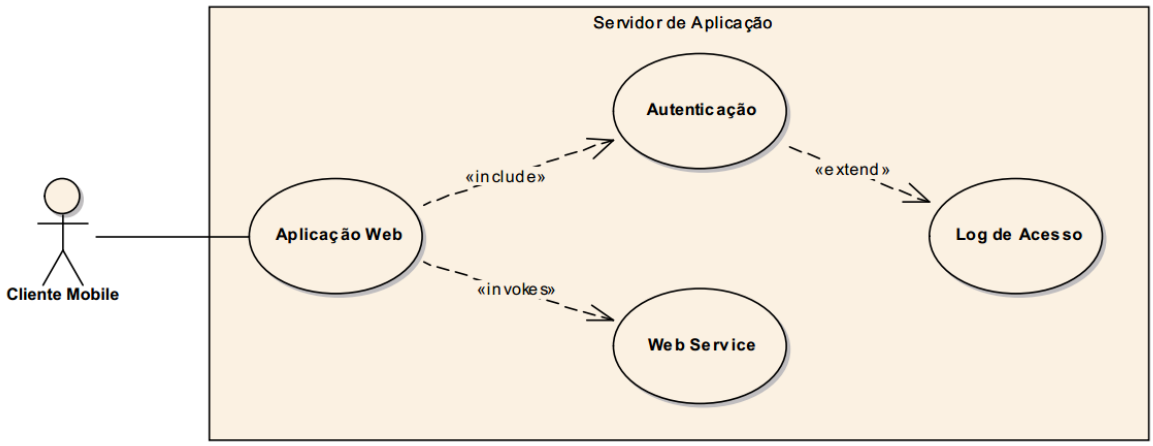

Figura 17 - Diagrama de Caso de Uso de acesso ao Web Service

4.2.2 Diagrama de Classes

Para facilitar a exposição, serão apresentadas neste diagrama as classes envolvidas na base da solução MVC[16] (*Model View Controller*) desenvolvida, bem como uma especialização de cada classe (Figura 18) para demonstrar a modelagem da entidade Cliente. As demais entidades a serem providas pelo serviço web seguem o mesmo padrão de modelagem, com seus atributos individuais.

[16] Paradigma de programação em que o problema é dividido em três camadas, sendo elas: *Model* (Modelo), *View* (Visão) e *Controller* (Controle). Sua principal finalidade é separar modelos de dados, *interface* de usuário e regras de negócio.

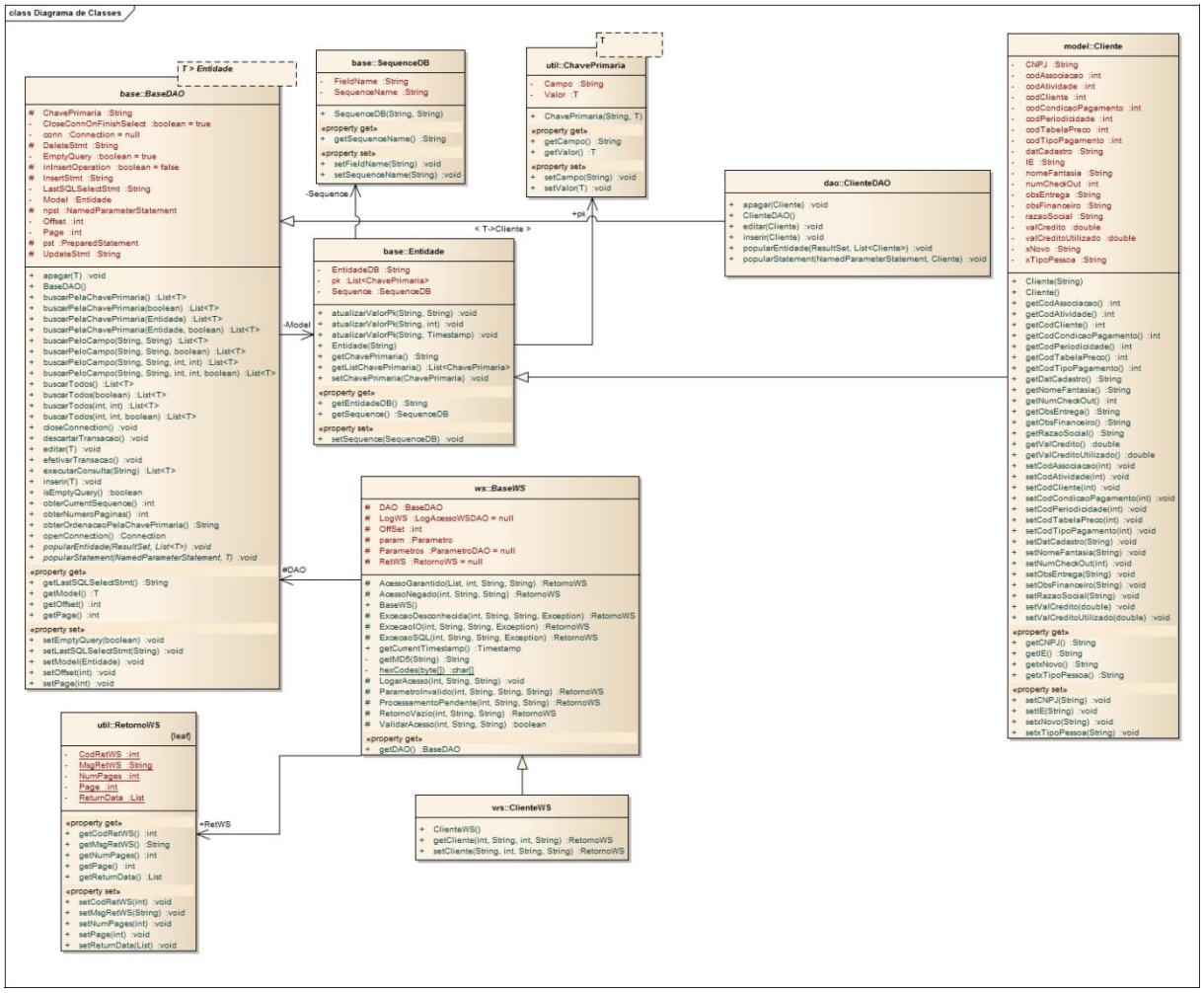

Figura 18 - Diagrama de Classes simplificado

4.2.3 Diagrama de Sequência

Uma vez que o diagrama de Caso de Uso apresenta uma visão limitada do sistema em si e que o Diagrama de Classes é uma representação estrutural, fez-se necessário o desenvolvimento de Diagramas de Sequência, para demonstrar o fluxo de comunicação entre os componentes do sistema, proporcionando assim ao desenvolvedor uma visão mais aguçada de como as partes devem interagir.

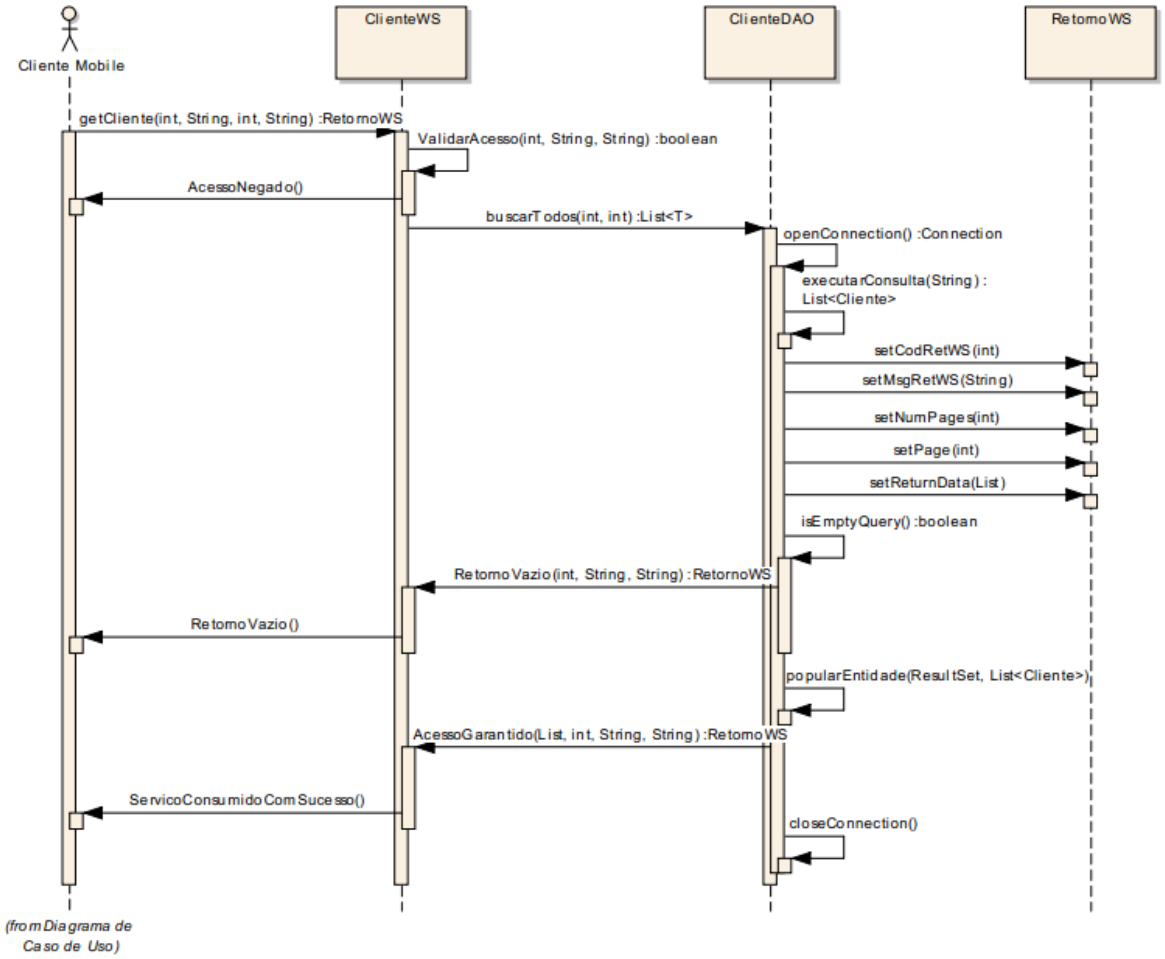

Figura 19 - Diagrama de Sequência da solução

4.3 DISCUSSÃO DOS RESULTADOS OBTIDOS

Com base na arquitetura proposta aplicada sobre o estudo de caso são apresentados os resultados considerando alguns aspectos, entre eles:

a) Ganho de desempenho

Contrariando o princípio básico da arquitetura encontrada no cenário atual da empresa, onde o acesso aos dados de sincronização por parte dos dispositivos móveis era feito por meio de conexão direta a uma base de dados, na arquitetura proposta esta conexão foi substituída pelo consumo de serviços web.

Ao contrário de uma conexão a uma base de dados, que remete ao tráfego de informações de transações, e outros dados pertinentes a garantia de integridade dos dados, o consumo ao serviço web resume-se ao download das informações pertinentes, uma vez que todo tratamento de integridade de dados e transações e realizado pelo servidor de aplicação

onde os serviços web estão publicados. Desta forma o fluxo de dados trafegados é reduzido, agilizando o processo de sincronização.

Outro ponto forte desenvolvido na arquitetura aqui proposta, que fez com que o desempenho da solução fosse incrementado substancialmente, refere-se a um mecanismo de paginação de dados. Este mesmo método utilizado por sites de busca como o Google, por exemplo, para demostrar os resultados das pesquisas, faz com que os dados sejam trafegados em fatias menores. Caso haja uma falha de conexão do dispositivo móvel com os serviços web durante a sincronização, na próxima tentativa de sincronização, os dados são trafegados a partir da página que estava sendo transmitida no momento da falha.

b) Robustez

Neste quesito a paginação de dados é também um ponto de destaque, pelo fato de garantir que a cada página transferida os dados sejam devidamente persistidos na base de dados, desta forma, a sincronização pode continuar da página em que parou em caso de uma retomada de falha de conexão.

Esta técnica garante a integridade das informações com a mesma segurança do cenário atual, e sem a necessidade de trafegar todos os dados de uma sincronização completa novamente.

c) Segurança

Além da segurança aplicada à regra de persistência de dados, garantindo a integridade dos dados persistidos tanto no dispositivo móvel, quanto no banco online, a restrição ao acesso dos serviços web também promove a segurança dos serviços e das informações manipuladas por eles.

Para possibilitar o acesso a qualquer serviço, o dispositivo móvel envia um código com o código e a senha do usuário do sistema daquele dispositivo, incrementado por uma chave conhecida tanto pelo serviço web quanto pelo dispositivo móvel transformados em um *hash* MD5[17] (*Message-Digest Algorithm*). O código do usuário, e seu *login* também são enviados como parâmetros para possibilitar o acesso ao serviço web.

Ao receber esses dados a aplicação de serviços web, recria o MD5 de acesso com base no *login* e senha do usuário (a senha é obtida do banco de dados online utilizando-se o

[17] *Hash* de criptografia unidirecional.

código de usuário enviado por parâmetro), e compara com o MD5 enviado pelo dispositivo móvel. Caso ambos sejam idênticos, o acesso ao serviço é garantido.

Existe ainda uma restrição de acesso quanto ao usuário estar temporariamente bloqueado, neste caso, mesmo que os códigos MD5 coincidam, o acesso é revogado, retornando uma mensagem informando o status do usuário.

d) Interoperabilidade

O conceito de interoperabilidade segundo Fonseca, Egenhofer e Borges (2000), pode ser entendido como a capacidade de um sistema possui de trocar recursos. Para tanto os recursos trocados tem que ser compreensíveis entre os sistemas que participam da troca.

Na arquitetura proposta a interoperabilidade entre as plataformas móveis e desktop, pode ser claramente notada pela transparência na troca de objetos entre ambos os sistemas, transparência esta alcançada pelo uso dos serviços web.

Embora no estudo de caso não tenham sido aplicadas outras plataformas móveis além do iOS®[18], com a arquitetura proposta, nenhuma alteração nos softwares existentes se faz necessária para a comunicação com outras plataformas, sejam elas móveis, para a web, ou até mesmo outros sistemas de plataforma *desktop*.

Embasado nos quesitos apresentados anteriormente nesta sessão, destacam-se pontos que no cenário atual eram considerados fracos, e, empregando a arquitetura proposta foram revertidos para pontos fortes.

Um exemplo mais acentuado, não mencionando a diferença na utilização de dispositivos com uma tela maior e com maior poder de processamento se comparados os iPads® aos smartphones executando Windows Mobile®, é a velocidade de sincronização alcançada por esta nova arquitetura.

Na solução anterior uma tarefa de sincronização envolvendo dois pedidos de compras a serem enviados para a base de dados no servidor da empresa, utilizando a conexão GPRS, levava em média mais de dez minutos para ser completada, utilizando esta nova solução a mesma tarefa pode ser concluída em segundos.

Outro problema que afetava o desempenho eram as quedas de conexão durante o processo de sincronização. Nestas situações o processo deveria ser reiniciado desde o início.

[18] Sistema operacional para dispositivos móveis desenvolvido e comercializado pela Apple® e instalado em produtos como iPhone® e iPad®.

Com a paginação de dados, a porção máxima de dados a serem perdidos em caso de queda de conexão é a contida na página atualmente sendo transmitida. Ao retomar o processo, o mesmo continua partindo desta página de dados.

O mais atrativo dos benefícios aos olhos dos usuários, e também dos responsáveis pela comercialização da solução, é em si a possibilidade de uso dos iPads. Com um alto poder de processamento, uma tela ampla, de fácil manuseio e com muitos recursos de acessibilidade, teve uma aceitação geral dos envolvidos muito mais favorável se comparada aos *smartphones*[19].

[19] Telefones celulares com funcionalidades extras, tais como, acesso a internet, aplicativos de escritórios, e-mails, etc.

5 CONSIDERAÇÕES FINAIS

O objetivo geral deste trabalho foi apresentar uma arquitetura que sustentasse a comunicação de dados entre sistemas de plataformas distintas de forma transparente, focando em requisitos de segurança, interoperabilidade, baixa suscetibilidade à erros e principalmente, o desempenho. Para atender aos objetivos buscou-se a aplicação da arquitetura de serviços web, devido ao fato de sua abordagem ampla, permitindo uma maior flexibilidade de desenvolvimento. Devido à variedade de soluções encontradas para o desenvolvimento desta arquitetura, foram estudadas as principais tecnologias, SOAP e RESTful. Com base na pesquisa realizada constatou-se um maior apelo pelo paradigma RESTful, devido a sua simplicidade e objetividade se comparado ao SOAP, o que para um dispositivo móvel com recursos limitados, tanto de processamento quanto de conexão, pesou na escolha.

Para validar a pesquisa realizada foi elaborado um estudo de caso, em que se criou uma aplicação de serviços web com o objetivo de promover a comunicação entre um sistema de vendas desenvolvido para plataforma móvel, mais especificamente o iPad, e um sistema ERP.

A arquitetura proposta contou com a inclusão de algumas particularidades importantes para o cumprimento dos objetivos propostos, e que também valorizaram a solução a ponto de ser adotada em ambiente de produção.

Para atender aos requisitos de segurança, foi utilizada uma variável contendo um *hash* MD5 com dados de acesso do usuário do sistema móvel, bem como seu código de usuário. Estes dados obrigatoriamente são passados a todos os serviços disponibilizados. A aplicação de serviços web, ao receber uma requisição, obtém do banco de dados as credenciais do usuário através do código também passado via URL. Uma vez localizados os dados do usuário é gerado o mesmo *hash* MD5 e comparado com o informado pelo cliente. O acesso é liberado somente se todos os dados coincidirem.

Outra particularidade da arquitetura aqui proposta foi à adoção da paginação de resultados retornados pela consulta aos serviços. Esta funcionalidade permitiu a diminuição dos erros, uma vez que a cada página de dados processada pelo dispositivo móvel os dados são validados evitando assim a inserção de grandes quantidades de informação até que fosse encontrada uma inconsistência. Isto permitiu ainda um ganho de desempenho pela facilidade de carregamento de cada página de dados individualmente, e pelo fato de poder continuar obtendo dados em caso de queda de conexão a partir da página que estava sendo transmitida no momento da queda.

A interoperabilidade, ou capacidade de integração de recursos computacionais heterogêneos foi alcançada pelo simples fato da adoção da arquitetura de serviços web na constituição do projeto, já que esta é uma das características da arquitetura, conseguida pelo método de serialização de objetos.

Contudo, este trabalho limitou-se apenas na apresentação das tecnologias envolvidas na arquitetura proposta, abordando as características intrínsecas de cada componente envolvido até o ponto relevante para a realização do estudo de caso. Ateve-se também a realidade da empresa na qual o projeto foi desenvolvido, e propôs-se a substituir a solução até então adotada, atendendo aos mesmos requisitos e possibilitando uma expansão facilitada de funcionalidades e plataformas.

Visando possíveis melhorias futuras nesta solução, menciona-se a integração de outras plataformas móveis, web, ou qualquer outra capacitada a implementar uma interface cliente de modo que se possa para consumir os serviços web desenvolvidos.

REFERÊNCIAS

ARROQUI, M. et al. RESTful Web Services improve the efficiency of data transfer of a whole-farm simulator accessed by Android smartphones. **Computers and Electronics in Agriculture**, 87, Setembro 2012.

BERNERS-LEE, T. The World Wide Web: Past, Present and Future. **W3C.org**, 1996. Disponivel em: <http://www.w3.org/People/Berners-Lee/1996/ppf.html>. Acesso em: 21 mar. 2012.

BERTRAM, J.; KLEINER, C. Secure Web Service Clients on Mobile Devices. **Procedia Computer Science**, 10, 2012.

BLAKE, M. B. et al. Binding Now or Binding Later: The Performance of UDDI Registries. **System Sciences 40th Annual Hawaii International Conference on**, Janeiro 2007.

BOX, D. O'Reilly XML.Com. **A Brief History of SOAP**, 04 Abril 2001. Disponivel em: <http://www.xml.com/pub/a/ws/2001/04/04/soap.html>. Acesso em: 19 fev. 2012.

CHAO, Y.; NENGCHENG, C.; LIPING, D. RESTFul based heterogeneous Geoprocessing workflow interoperation for Sensor Web Service. **Computers & Geosciences**, Outubro 2012.

CURBERA, F. et al. Unraveling the Web services web: an introduction to SOAP, WSDL, and UDDI. **Internet Computing, IEEE**, v. 6, n. 2, Março-Abril 2002.

DUMBILL, E.; JHONSTON, J.; LAURENT, S. S. **Programming Web Services with XML-RPC**. Sebastopol: O'Reilly, 2001.

EVANS, M. **The Evolution of the Web - From Web 1.0 to Web 4.0**. School of Systems Engineering - University of Reading. [S.l.]. 2008.

FIELDING, R. et al. Hypertext Transfer Protocol -- HTTP/1.1. **The Internet Engineering Task Force (IETF)**, Junho 1999. Disponivel em: <http://www.ietf.org/rfc/rfc2616.txt>. Acesso em: 01 fev. 2013. RFC-2616.

FIELDING, R. T. **Architectural Styles and the Design of Network-based Software Architectures**. University of California. Irvane. 2000.

FILHO, O. F. F. **Serviços Semânticos: Uma abordagem RESTful**. Escola Politécnica da Universidade de São Paulo. São Paulo, p. 103. 2009.

FONSECA, F.; EGENHOFER, M.; BORGES, K. A. V. Ontologias e Interoperabilidade Semântica entre SIGs. **Interoperability in GIS - A Joint NSF/CNPq Project** , Orono, 2000.

FONSECA, R.; SIMÕES, A. **Alternativas ao XML : YAML e JSON**. Universidade do Minho. [S.l.], p. 14. 2007.

GONSALVES, A. **Beginning Java EE 6 Plataform with GlassFish 3:** From Novice to Professional. New York: Apress, 2009.

GOOGLE INC. A well earned retirement for the SOAP Search API - The Official Google Code blog. **Google Code**, 2009. Disponivel em: <http://googlecode.blogspot.com.br/2009/08/well-earned-retirement-for-soap-search.html>. Acesso em: 21 mar. 2012.

HADLEY, M. Web Application Description Language. **W3C Member Submission**, 2009. Disponivel em: <http://www.w3.org/Submission/wadl/>. Acesso em: 23 fev. 2012.

HAMAD, H.; SAAD, M.; ABED, R. Performance Evaluation of RESTful Web Services for Mobile Devices. **International Arab Journal of e-Tecnology**, Palestina, Janeiro 2010.

HAMMER-LAHAV, E. RFC 5849 - The OAuth 1.0 Protocol. **Internet Engineering Task Force (IETF)**, 2010. Disponivel em: <http://tools.ietf.org/html/rfc5849>. Acesso em: 21 mar. 2012.

HENDLER, J. Web 3.0 Emerging. **Computer**, 42, n. 1, Janeiro 2009. 111-113.

HONG, Y. A Resource-Oriented Middleware Framework for Heterogeneous Internet of Things. **Cloud and Service Computing (CSC), 2012 International Conference on**, p. 12-16, 22-24, Novembro 2012.

KAMALELDIN , M.; DUMINDA, W. Performance Analysis of Web Services on Mobile Devices. **Procedia Computer Science**, v. 10, p. 744-751, 2012.

LEMOS, A. Comunicação e práticas sociais no espaço urbano: as características dos dispositivos híbridos móveis de conexão multirredes (DHMCM). **Comunicação, mídia e consumo**, v. 4, n. 10, 2007.

MACHADO, A. et al. Utilização de Dispositivos Móveis, Web Services e Sotfware Livre no Monitoramento Remoto de Pacientes. **Congresso Brasileiro de Informática em Saúde**, Campos do Jordão, v. 1, p. 1-6, 2008.

MCFADDIN, S. et al. Modeling and Managing Mobile Commerce Spaces using RESTful Data Services. **IEEE Computer Society**, 2008.

O'REILLY, T. Design Patterns and Business Models for the Next Generation of Software. **O'REILLY Spreading the knowledge of innovators**, 2005. Disponivel em: <http://oreilly.com/web2/archive/what-is-web-20.html>. Acesso em: 15 abr. 2012.

ORT, E. The Source - Sum microsystems. **Service Oriented Architecture and Web Services: Concepts, Tecnologies, and tools**, 2005.

PILIOURA, T. et al. Using Web Services for supporting the users of wireless devices. **ScienceDirect**, Athenas, 27 Junho 2005.

PRESSMAN, R. S. **Engenharia de Software**. 6. ed. [S.l.]: McGraw-Hill , 2006.

RATINIMITTUM, W.; PIROMSOPA, K. An implementation of RESTful-based Scalable File System. **Computer Science and Software Engineering (JCSSE), 2012 International Joint Conference on**, 2012 maio 30. 136-141.

RICHARDSON, L.; RUBY, S. **RESTful Web Services**. [S.l.]: O'Reilly, 2007.

SHOMOYITA, J.; RALPH, D. Using a Cloud-Hosted Proxy to support Mobile Consumers of RESTful Services. **Procedia Computer Science**, v. Volume 5, p. 625-632, 2011.

SOMMERVILLE, I. **Software Engineering**. 8. ed. [S.l.]: [s.n.], 2007.

TIDWELL, D.; SNELL, J.; KULCHENKO, P. **Programing Web Services with SOAP**. 1. ed. [S.l.]: O'Reilly, 2001. 216 p.

TOTTY, B. et al. **HTTP - The Definitive Guide**. 1. ed. [S.l.]: O'Reilly Media, Incorporated, 2002.

TURTSCHI, A. et al. Chapter 11 - Web Services. **C#.NET Web Developer's Guide**, Burlington, p. 575-668, 2002.

W3C. Web Services Architecture. **W3C**, 2004. Disponivel em: <http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/#whatis>. Acesso em: 12 set. 2012.

ZHAO, W. Building Highly dependable Wireless Web Services. **Journal of Electronic Commerce in Organizations**, p. 1-16, 10 a 12 2010.