UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

DEPARTAMENTO ACADÊMICO DE INFORMÁTICA

CURSO DE ESPECIALIZAÇÃO EM TECNOLOGIA JAVA

Paulo Sergio Bernardo Pinto

WebService – A Realidade da Computação Distribuída Como Ferramenta de Integração entre Tecnologias.

Trabalho de Conclusão de Curso

Curitiba
2011

Paulo Sergio Bernardo Pinto

WebService – A Realidade da Computação Distribuída Como Ferramenta de Integração entre Tecnologias.

Monografia de Especialização apresentada ao Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito parcial para obtenção do título de "Especialista em Tecnologia Java".
Orientador: Prof. Caio Nakashima
Co-orientador: Prof. Paulo Bordin

Curitiba
2011

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter me dado a força e a perseverança necessária para que este objetivo fosse alcançado. Agradeço aos meus filhos a minha esposa por terem sido meu apoio e peça fundamental nesse difícil caminho que foi trilhado. Agradeço principalmente a aqueles que disseram que estudar era inútil e que hoje ficaram pelo caminho, pois hoje sei que eu estava certo.É com grande satisfação que dou o meu "Muito Obrigado!".

RESUMO

Este trabalho apresenta um estudo sobre a utilização de Web Services com a tecnologia Java, demonstrando a capacidade dos trabalhos que podem ser disponibilizados e compartilhados sem a necessidade de controle ou informações dos clientes que a utilizam, sem perder os pré-requisitos de segurança, os exemplos apresentados foram desenvolvidos em códigos JAX-WS (Java API for XML-Web Services).Um web service é, uma aplicação distribuída, cujos componentes podem ser aplicados e executados em dispositivos e linguagens distintas.

FIGURAS

Figura 01:Arquitetura de um típico Web Service baseado em SOAP 15

Figura 02: Processo geral para estabelecer um Web Service 16

Figura 03: Mensagem SOAP para a invocação do método RetColab com os argumentos 1,1,29750 17

Figura 04: Mensagem SOAP de resposta ao pedido apresentado na Figura 3 17

Figura 05: Arquitetura de um Web Services tradicional baseado em Rest. 18

Figura 06: Organização do modelo SOAP e REST. 20

Figura 07: Demonstração da aplicação. 32

Figura 08: Diagrama de caso de uso do web service de retorno de dados cadastrais UML. 34

Figura 09: Diagrama de caso de uso do web service de retorno de dados cadastrais. 35

Figura 10: Diagrama de implantação 36

Figura 11: Codigo do exemplo da aplicação que utiliza a chamada do web service por código java 37

Figura 12: Codigo do exemplo do XML gerado pela aplicação que utiliza a chamada do web service por código java 39

Figura 13: Codigo do exemplo da aplicação que consome web service 40

Figura 14: Codigo do exemplo da aplicação que consome web service por código Java, jse, direto no prompt do sistema operacional Windows 42

Figura 15: Codigo do exemplo da aplicação que consome web service por código Java, jee 42

Figura 16: Imagem do retorno da chamada web do consumo de web service. 43

SUMÁRIO

1. INTRODUÇÃO 8

1.1. Justificativa 10

1.2. Descrição Dos Capitulos 11

2. REFERENCIAL TEÓRICO 12

2.1. Sistemas Distribuidos 12

2.2. Desempenho e Escalabilidade 12

2.3. Conectividade e Segurança 12

2.4. Confiabilidade e Tolerância a Falhas em Sistemas Distribuídos 13

2.5. Transparência 13

2.6. Comunicação em Sistemas Distribuídos 14

2.7. Modelos de Computação Distribuída 14

2.8. Web Service 14

2.9. SOAP 15

2.10. REST 18

2.11. Infográfico de Comparação entre *SOAP* E *REST* 20

3. DEFINIÇÃO DO MODELO ESCOLHIDO 21

3.1. Web Service, Discurso Detalhado 21

3.2. Benefícios de serviços web 23

3.3. Evolução de tecnologias e produtos 24

3.4. Segurança 24

3.5. Reusabilidade, Disponibilidade e Escalabilidade 26

3.6. Cenários de serviços web 26

3.7. Interação com Padrões de Negócio 27

3.8. Integrando com Sistemas de Informação Existentes 28

3.9. Alcançando diversos clientes 28

4. ESTUDO DE CASO 29

4.1. Descrição da Aplicação Exemplo 31
4.2. Diagrama de Caso de Uso 33
4.3. Especificação do Projeto 34
4.4. Tecnologia. 34
4.5. Arquitetura 35
4.6. Diagrama de implantação 35
4.7. Construindo e Consumindo um Web Services 36
4.8. Implementando um Web Services. 36
4.9. Consumindo um Web Wervices 41
5. CONCLUSÃO **44**
6. REFERÊNCIAS BIBILIOGRÁFICAS **46**

1. INTRODUÇÃO

A necessidade de informação, a evolução dos equipamentos com o fácil acesso a novas tecnologias criam o advento do imediatismo, tornando até mesmo funções antes corriqueiras como pedir uma pizza, um gatilho para se utilizar ferramentas tecnológicas e ou serviços antes inexistentes, gerando profunda angústia quando não executada a contento. E com este cenário notamos que a computação toma rumos onde aplicações deixam de ser apenas locais, ou seja, concentradas em uma única estação ou servidor, e passam a ser distribuídas geograficamente em um ambiente qualquer, com isto centrais de atendimento podem estar fisicamente em qualquer localização geográfica e atender a contento as necessidades, um exemplo:

- É o sistema utilizado pela rede de lanchonetes Habib's, onde os pedidos são realizados através de uma central (0800), e após designados a loja mais próxima do endereço que solicitou.

Nestas aplicações, há necessidade de se ter um maior recurso computacional com um custo baixo, motivando o uso da computação distribuída nos últimos anos, resultando em um aumento de pesquisa nesta área. Com o aumento do uso de redes de computadores e a sofisticação dos ambientes multitarefa torna-se inquestionável a importância dos mecanismos de otimização de computação. A utilização de técnicas de programação distribuída visa cumprir metas importantes baseadas em funcionalidades cujo objetivo é prover soluções a fim de proporcionar um melhor uso da capacidade computacional, propiciando vantagens como a interoperabilidade, o desempenho, a escalabilidade, a conectividade, a confiabilidade, a transparência, a tolerância à falhas e segurança em ambientes heterogêneos e homogêneos. Assim, cresce o número de aplicações que demandam acesso de forma integrada, uniforme em tais ambientes[1].

Comentando-se sobre aplicações hoje em dia é praticamente certo que o assunto acabará em web. E com essa visão, o trabalho foi direcionado para discutir uma tecnologia de comunicação, entre aplicações, que abstraísse em qual plataforma as aplicações estão sendo executadas ou em quais linguagens de programação foram desenvolvidas. Dentro deste contexto, este trabalho

apresenta um estudo sobre a utilidade dos Web Services (WS) ou Serviços Web (SW), para os processos de integração entre tecnologias e sistemas.

Com isto buscamos modelos de computação distribuída, conceituando e apresentando exemplos de suas características focando nos processo com Web Services.

Com isto buscamos modelos de computação distribuída, conceituando e exemplificando suas características focando nos processo com Web Services.

Além disso, um estudo de caso foi definido, com o intuito de exemplificar o funcionamento e a troca de informações entre aplicações clientes e os serviços propriamente ditos, além da interoperabilidade, que é uma das principais características dos modelos de computação distribuída. Esse estudo de caso é baseado em um cenário em que a computação distribuída é fortemente usada, simulando uma transferência entre informações entre sistemas com caraterísticas distintas. Serão utilizados um Sistema Integrado de Gestão Empresarial (SIGE), em inglês *Enterprise Resource Planning* (ERP) desenvolvido pela Senior Sistemas o sistema de Gerenciamento de Recursos Humanos Vetorh e o Sistema de ERP Sapiens, e outros sistemas desenvolvidos internamente.

Com o desenvolvimento da Internet e das tecnologias paralelas, surgiram novas abordagens ao problema da interoperabilidade entre sistemas de informação. A ubiquidade da "rede das redes", "indiferente" a fronteiras empresariais ou tecnológicas, surge como uma alternativa para a camada de comunicação. Por outro lado, a evolução tecnológica e a adoção generalizada da eXtensible *Markup* Language (XML), como linguagem de estruturação de documentos, veio permitir o desenvolvimento de novas arquiteturas, atualmente designadas de Serviços Web (SW)[2].

Os Serviços Web representam uma tecnologia emergente que tem sido alvo de atenção por parte de toda a indústria de computadores[3]. Empresas como a Microsoft, IBM e Sun têm apostado e investido neste conceito. As primeiras iniciativas remontam a 1998, com o trabalho desenvolvido pela Microsoft em parceria com a Userland Software, que resultou numa primeira publicação do protocolo SOAP[4]. A partir de 2001, o desenvolvimento de especificações passou a ser realizado em consórcios empresariais, como o W3C e a OASIS. De uma forma simplificada, os Serviços Web procuram facilitar a integração de

aplicações distribuídas utilizando XML sobre a World Wide Web (WWW ou Web). No entanto, de um ponto de vista prático, as estratégias para a implementação deste conceito divergem, resultando em alguma ambiguidade a este nível. Não existe uma noção clara do que é a implementação de um Web Service. Implementações com diferenças importantes ao nível da arquitetura e das tecnologias utilizadas têm sido catalogadas como "Web Service". Um outro aspecto que importa referir é o fato de, associado ao trabalho desenvolvido pelos consórcios empresariais, terem sido produzidas inúmeras especificações. Em finais de 2004, existiam mais de 30 especificações (p.e*.:* WSFederation*,* WSDL, BPEL4WS,WS-I Basic Profile) relativas à visão dos Serviços Web proposta pelos principais consórcios. Este aspecto tem contribuído para um maior afastamento entre os programadores, que desenvolvem as implementações reais, e as especificações propostas pelas grandes empresas. Com base no estudo feito e no cenário do estudo de caso (ver capítulo 4), foi escolhido um modelo específico (web service) para o desenvolvimento. Para tanto, um serviço e uma aplicação cliente foram desenvolvidos na linguagem de programação *Java*, e com o intuito de mostrar a interoperabilidade dos serviços web, outra aplicação cliente foi desenvolvida na linguagem Delphi, apenas como exemplo do que se pode atingir.

1.1. Justificativa.

As necessidades das empresas e a crescente demanda por aplicações distribuídas no cenário do mercado nacional, os desafios e a inovação que serviços web proporcionam às empresas de software, acrescidos da vontade de conhecer a fundo uma tecnologia nova, tomando como base uma linguagem de desenvolvimento nova, como o *Java* , para o estudo aprofundado da referida tecnologia nos motivou a realizar este estudo. Para atender as necessidades de disponibilizar informações de diversos setores da empresa, para diversos departamentos, sem a necessidade, sabermos quem e como iriam buscar estas informações, ou qual ferramenta de desenvolvimento seria utilizada ou linguagem de programação seria desenvolvida o cliente destes web services. Os sistemas utilizados na produção necessitavam informações cadastrais do funcionário: departamento, turno, nome completo,

etc. Os sistemas da área financeira, estavam preocupados com o nível de acesso dos colaboradores mediante o cargo hierárquico, etc. O método de desenvolvimento foi definido através da aplicações de prioridades e necessidade de cada cliente a ser desenvolvido e a sequencia de desenvolvimento foi definido pela diretoria de TI juntamente com as áreas clientes, chegando a sequencia desejada pela empresa.

1.2. Descrição dos Capítulos.

O capítulo 2 apresenta um referencial teórico, descrevendo o conceito de sistemas distribuídos e alguns modelos de computação distribuída com suas principais características, exemplos de aplicação, vantagens e desvantagens. Além disso, apresenta também uma comparação entre os modelos estudados.

O capítulo 3 apresenta a definição de um modelo de computação distribuída (serviços web), dentre as estudadas, para um estudo mais aprofundado, mostrando suas características, benefícios e outras peculiaridades relacionadas.

O capítulo 4 apresenta um estudo de caso, baseado em um cenário onde a computação distribuída é usada com ênfase. Esse estudo mostra a descrição de uma aplicação exemplo, bem como o seu desenvolvimento. Além disso, caracteriza uma demonstração prática da aplicação desenvolvida.

O capítulo 5 contém a conclusão do trabalho, mostrando os pontos que foram abordados durante o mesmo. Além disso, são apresentadas algumas sugestões de trabalhos futuros.

O capítulo 6 mostra as referências que foram utilizadas ao longo do trabalho.

2. REFERENCIAL TEÓRICO

2.1. Sistemas Distribuídos.

A comunicação remota via rede, originalmente reservada para grandes instalações de computadores e ambientes acadêmicos, foi amplamente difundida com o advento da internet (rede mundial de computadores). Em um sistema distribuído, computadores independentes cooperam via rede de modo que pareçam uma máquina local. Aplicações de sistemas distribuídos podem executar código em máquinas locais e remotas e compartilhar dados, arquivos, e outros recursos entre máquinas. Sistemas distribuídos quase sempre surgem da necessidade de melhorar a capacidade, a confiabilidade de uma única máquina e atender uma grande base de usuários. Algumas das vantagens de se utilizar a tecnologia de sistema distribuído é que se pode atingir um alto nível de desempenho por um custo menor do que um sistema único[5].

Os sistemas distribuídos são atribuídos de vantagens em relação a sistemas centralizados como desempenho e escalabilidade, conectividade e segurança, confiabilidade, tolerância a falhas e transparência. Vantagens essas que são apresentadas ao longo deste tópico.

2.2. Desempenho e Escalabilidade.

Em um sistema centralizado, um único servidor trata todas as requisições de usuários. Com um sistema distribuído, as requisições podem ser enviadas a diferentes servidores aumentando o desempenho. Escalabilidade permite que um sistema distribuído cresça sem afetar as aplicações e os usuários existentes[6] .

2.3. Conectividade e Segurança.

Um sistema distribuído pode fornecer acesso sem descontinuidade e recursos através da rede[7]. Se o recurso for um processador, o sistema distribuído deverá permitir que tarefas sejam executadas em qualquer máquina. Se o recurso for um sistema de arquivos globalmente compartilhado, usuários remotos poderão acessá-lo como acessariam um sistema de arquivos local, privado. Conectividade em sistemas distribuídos requer protocolos de

comunicação de estado a fim de manter uma operação eficiente, devem fornecer interfaces comuns a todos os computadores do sistema.

Sistemas distribuídos devem permitir que apenas usuários autorizados acessem recursos e garantir que a informação transmitida pela rede somente possa ser lida pelos recipientes pretendidos[8].

2.4. Confiabilidade e Tolerância a Falhas em Sistemas Distribuídos.

Sistemas distribuídos implementam tolerância a falhas fornecendo replicação de recursos através do sistema. A replicação oferece aos usuários maior confiabilidade e disponibilidade em relação a implementações de máquinas isoladas. Sistemas distribuídos devem ter um software que detecte e reaja a falhas no sistema, fornecer mecanismos para assegurar a consistência entre informações de estado de máquinas diferentes, e precisam estar equipados para reintegrar recursos que falharam, logo que sejam reparados[9] .

2.5. Transparência.

Transparência em sistemas distribuídos é ocultar dos usuários seus aspectos distribuídos[10]. Esses sistemas devem implementar transparência de localização, ocultando a localização de recursos no sistema distribuído daqueles que estão tentando acessá-lo, permitindo acesso a arquivos remotos como se fossem locais. Essa característica pode ser alcançada por replicação ou por ponto de verificação/recuperação de sistema onde a replicação fornece vários recursos que executam a mesma função e, ao usar pontos de verificação, periodicamente serão armazenados informações de estado de um objeto, como um processo.

Além dessas, também devem ser implementadas transparências de migração ou transparência de realocação. A transparência de migração se preocupa em mascarar a movimentação de um objeto de uma localização para outra no sistema. Já a transparência de realocação mascara a realocação de um objeto por meio de outros objetos que se comunicam entre eles.

Por fim, sistemas distribuídos devem implementar transparência de transação, permitindo que um sistema obtenha consistência mascarando a coordenação entre um conjunto de recursos[11].

2.6. Comunicação em Sistemas Distribuídos.

Gerenciar a comunicação entre computadores é um desafio em sistemas distribuídos, onde se deve estabelecer interoperabilidade entre computadores e aplicações heterogêneas[12].

Para que um sistema, ao ser projetado, alcance as características de um sistema distribuído, esse deve ser desenvolvido tendo por base algum modelo de computação distribuída, e como exemplo desses modelos, tem-se: chamada remota de procedimento RPC (acrônimo de Remote Procedure Call), RMI (Remote Method Invocation),P2P (Pelt to pelt), CORBA, Grid, Cluster e Web Services, que são apenas alguns, de vários existentes. Entre esses modelos nosso foco principal será o de Web Services, a qual pode ser dividido em dois grupos, baseados em SOAP (Simple Object Access Protocol) e de estilo REST (Representational state transfer), um serviço baseado em SOAP fornecido através de HTTP é um caso especial de serviço de estilo REST[13].

2.7. Modelos de Computação Distribuída.

Os modelos apresentados a seguir podem ser tomados como base para o desenvolvimento de aplicações distribuídas. É lógico que o modelo a ser escolhido depende do objetivo da aplicação a ser desenvolvida.

2.8. Web Service.

O termo web service apesar de vários significados, pontuando algumas características típicas de Web Services será o suficiente para nos introduzir na codificação de um web service e um cliente, também conhecido como solicitante ou consumidor, um web service é um tipo de aplicação para a web, uma aplicação tipicamente oferecida através de HTTP (Hyper Text Transport Protocol) é, então, uma aplicação distribuída, cujos componentes podem ser aplicados e executados em dispositivos distintos. Como exemplo um web service de escolha de ações pode consistir de diversos componentes de

código, cada um armazenado em um servidor de grau de negócios separado, e o web service pode ser consumido nos mais diversos dispositivos,em computadores pessoais PCs, celulares, tablet, etc.

Web Services podem ser divididos em dois grupos, baseados em SOAP e de estilo REST.

2.9. SOAP.

A distinção não é precisa porque, um serviço baseado em SOAP fornecido através de HTTP é um caso especial de um serviço de estilo REST. SOAP (SIMPLE OBJECT ACCESS PROTOCOL) protocolo de acesso a objetos simples, é um protocolo para troca de informações estruturadas, ele se baseia em XML (Extensible Markup Language), é um protocolo de arquitetura orientada a serviço SOA (Service Oriented Architecture). No momento, SOAP é apenas um dialeto XML (Extensible Markup Language), no qual documentos são mensagens. Em Web Services baseados em SOAP, SOAP é a infraestrutura menos vista. Por exemplo em um típico cenário, chamado de troca de mensagem pedido/reposta em inglês, MEP(Message Exchange Parttern) solicitação/resposta, a biblioteca SOAP fundamental do cliente envia uma mensagem SOAP como uma solicitação de serviço, e a biblioteca SOAP subjacente do web service envia outra mensagem SOAP como a resposta correspondente ao serviço.

SOAP é o envelope das mensagens, contendo regras de codificação, possui estrutura para definir a forma de comunicação, estrutura para tratar erros, estrutura de extensão que permite evolução.

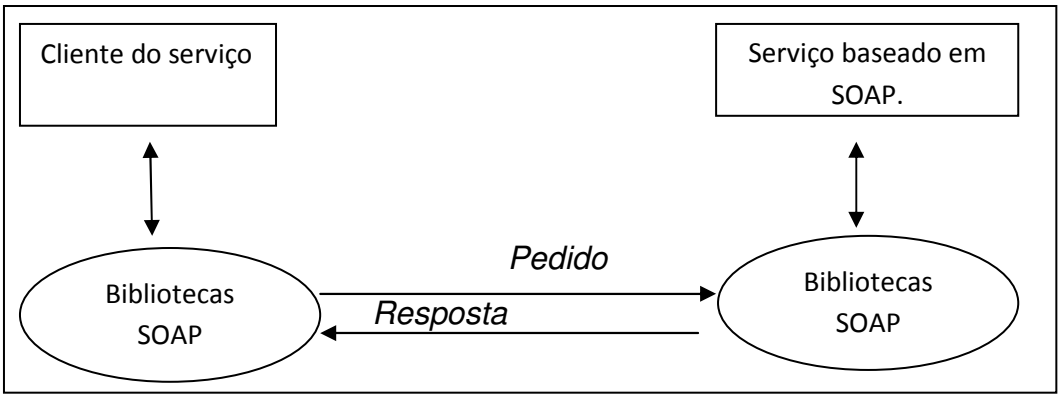


Figura 1.Arquitetura de um típico web service baseado em SOAP.

Fonte:[Java Web Services – Implementando - Martin Kalin]

A Figura 1. representa o modelo apresentado pelo W3C relativo ao processo geral para estabelecimento de um Web Service[14].

O conceito de Web Service está associado á disponibilização de funcionalidades através de interfaces programáveis via Internet. Na definição apresentada pelo W3C (Consórcio *World Wide Web*), é referido o uso de URL (*Uniform Resource Locator*), HTTP e XML em "conjunto com outras normas relacionadas com a Web". No entanto, apesar do recursoas designados tecnologias Web, o estilo arquitetural proposto para os WS não segue aquele desenvolvido para a WWW.

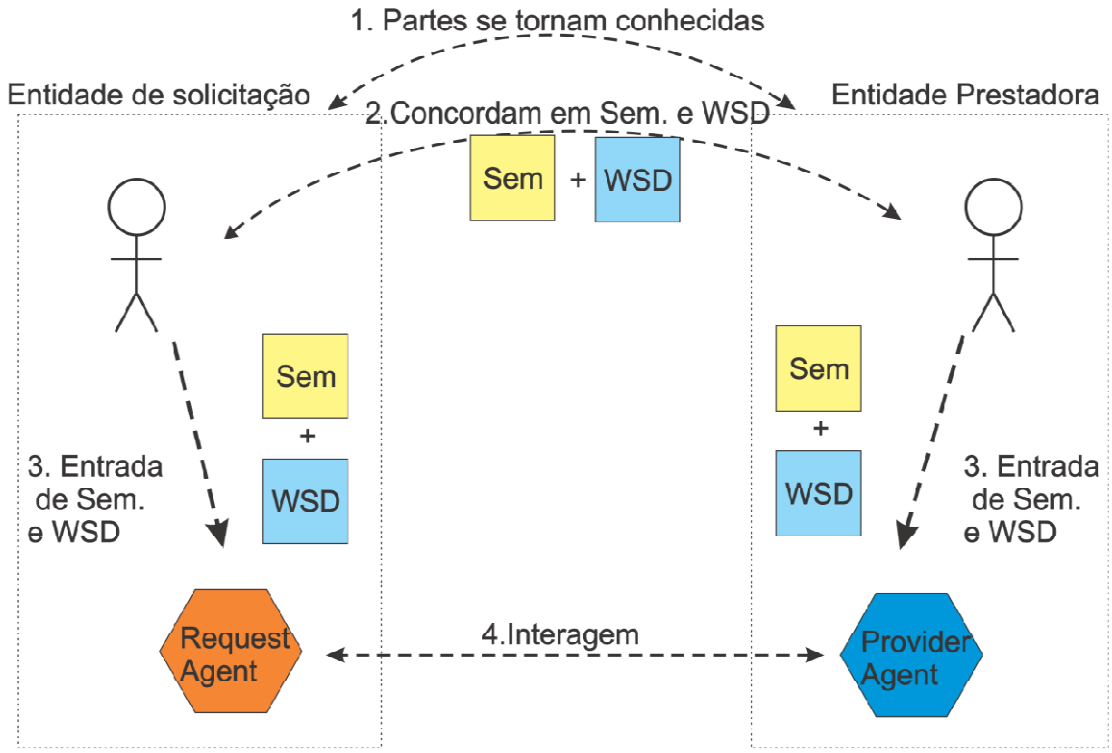


Figura 2. Processo geral para estabelecer um Web Services[15]
Fonte:[Web Services Architecture W3CWorking Group Note*]*

Através da Figura 2. é possível identificar as fases de execução do web service:(1) o cliente e o servidor tornam-se conhecidos; (2) o cliente e o servidor estabelecem um acordo em relação à descrição e semântica do serviço que vai gerir a interação entre os agentes; (3) é realizado o procedimento de implementações de acordo com a descrição e as referencias acordadas; (4) o cliente e o servidor trocam mensagens que concretizam a

interação entre as partes. De acordo com a visão tradicional para os Serviços Web, a fase de interação é implementada com recurso a troca de mensagens *SOAP*.

A Figura 3. apresenta uma mensagem SOAP correspondente a invocação de um procedimento.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Header/>
    <S:Body>
        <ns2:RetColab xmlns:ns2="http://ws/">
            <name>1</name>
            <arg1>1</arg1>
            <arg2>29750</arg2>
        </ns2:RetColab>
    </S:Body>
</S:Envelope>
```

Figura 3. Mensagem SOAP para a invocação do método `RetColab` com os argumentos 1,1,29750.

Na Figura 4. apresenta-se a resposta ao pedido. O código aqui apresentado é uma versão simplificada daquele que seria necessário num cenário real.

```
<?xml version="1.0" encoding="UTF-8"?>
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
    <S:Body>
        <ns2:RetColabResponse xmlns:ns2="http://ws/">
            <return>
                <nomloc_gestor>ADM SÃ„O JOSE</nomloc_gestor>
                <r006esc_nomesc>7:50-11:50 E 13:00-
17:48</r006esc_nomesc>
                <r018ccu_nomccu>FRETE PROPRIO - SC</r018ccu_nomccu>
                <r024car_titred>AJ.MOTORISTA 3</r024car_titred>
                <r034fun_apelido>XXXXXXXXXXXXXX XXXX</r034fun_apelido>
                <r034fun_codcar>533</r034fun_codcar>
                <r034fun_codccu>302532</r034fun_codccu>
                <r034fun_codesc>300</r034fun_codesc>
                <r034fun_codfil>2</r034fun_codfil>
                <r034fun_codtma>1</r034fun_codtma>
                <r034fun_datnas>1949-02-24 00:00:00.0</r034fun_datnas>
                <r034fun_nomfun>NONO NONO NONO</r034fun_nomfun>
            </return>
        </ns2:RetColabResponse>
    </S:Body>
</S:Envelope>
```

Figura 4. Mensagem SOAP de resposta ao pedido apresentado na Figura 3

O SOAP não define, nem está limitado, a um mecanismo de transporte específico. A especificação permite a integração com diversos protocolos para a transferência das mensagens, por exemplo: HTTP, (*Simple Mail Transfer Protocol*) SMTP e MQSeries. Atualmente, a generalidade das implementações utilizam o protocolo HTTP, em particular o método POST, para a troca de mensagens SOAP. Este cenário é ilustrado na Figura 5.

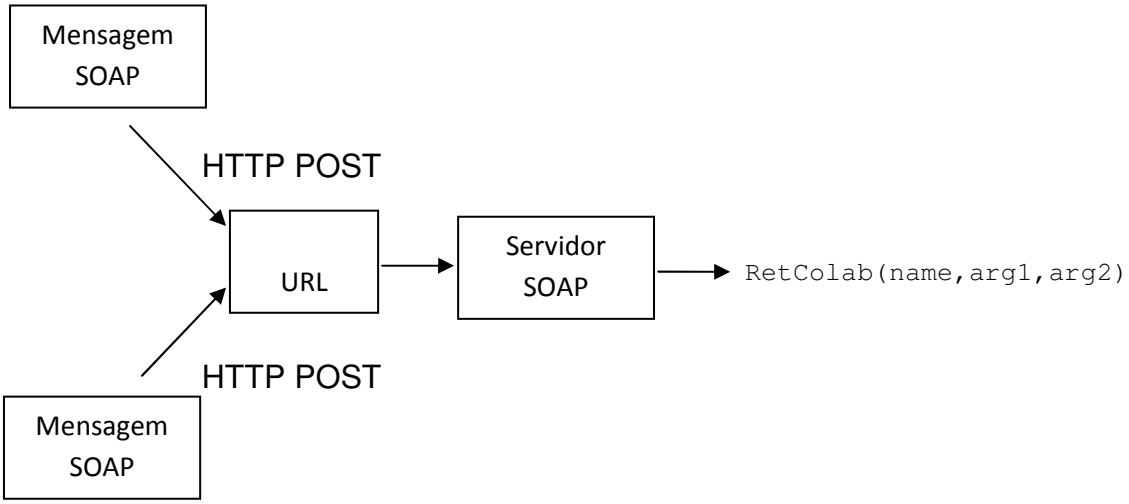


Figura 5. Arquitetura de um Web Services tradicional baseado em Rest.
*Fonte:[*Costello, R. L.: REST (Representational State Transfer).*]*

Não é um requisito da especificação a utilização da mesma URL para o envio de todas as mensagens. No entanto, esta é uma estratégia comum entre os promotores do SOAP.

2.10. REST

A REST Transferência de Estado Representacional (Representational State Transfer) ou somente (REST) é uma técnica de engenharia de software para sistemas hipermídia distribuídos como a *World Wide Web*. O termo se originou no ano de 2000, em uma tese de doutorado[16](PHD) sobre a web escrita por Roy Fielding, um dos principais autores da especificação do protocolo HTTP que é utilizado por sites da internet.

> A REST (Transferência do Estado Representacional) é pretendida como uma imagem do design da aplicação se comportará: uma rede de websites (um estado virtual), onde o usuário progride com uma aplicação selecionando as ligações (transições do estado), tendo como resultado a página seguinte (que representa o estado seguinte da aplicação) que está sendo transferida ao usuário e apresentada para seu uso. (Dr.Roy Fielding – 1985)

O termo *REST* se referia, originalmente, a um conjunto de princípios de arquitetura de sistemas de informações distribuídos, primeiramente chamada "HTTP Object Model", atualmente usada no sentido mais amplo para descrever qualquer interface web simples que utiliza XML e HTTP, possui protocolos baseados em padrões de trocas de mensagem como o protocolo de Web Services SOAP. É possível projetar sistemas de serviços web de acordo com o estilo arquitetural REST descrito por Fielding.O modelo REST utiliza um conjunto de interfaces genéricas para promover Interações sem estado (*stateless*) através da transferência de representações de recursos, em vez de operar diretamente sobre esses recursos. O conceito de recurso é a principal abstração deste modelo. Entre as restrições definidas pelo REST, destacam-se três e refere-se a aplicação no caso concreto da Web:

– Identificação global. Todos os recursos são identificados através do mesmo mecanismo global. Independentemente do contexto de aplicação, tecnologia ou implementação concreta, cada recurso disponível na Web é identificado utilizando um URL[17] .

– Interfaces uniformes. A interação entre os agentes é feita com recurso ao protocolo HTTP[18], utilizando um conjunto de métodos pré-definidos:
GET, POST, PUT e DELETE.

– Interações *stateless*. O estado concreto de uma aplicação Web é mantido com base no estado de um conjunto de recursos. O servidor conhece o estado dos seus recursos mas não mantém informação sobre as sessões dos clientes.

Um conceito importante em REST é a existência de recursos, que podem ser usados utilizando um identificador global (um Identificador Uniforme

de Recurso) para manipular estes recursos, os componentes da rede (clientes e servidores) se comunicam através de uma interface padrão (HTTP) e trocam representações de recursos (os arquivos ou ficheiros são recebidos e enviados) - é uma questão polêmica e gera muito debate, sem a distinção entre recursos e suas representações é demasiado utópica para seu uso prático na rede, onde é popular na comunidade RDF.

O pedido pode ser transmitido por qualquer número de conectores (por exemplo clientes, servidores, caches, etc.) mas não poderá ver mais nada do seu próprio pedido (conhecido com separação de capas, outra restrição do REST, que é um princípio comum com muitas outras partes da arquitetura de redes e da informação). Assim, uma aplicação pode interagir com um recurso conhecendo o identificador do recurso e a ação requerida, não necessitando conhecer se existem caches, proxys ou qualquer outra coisa entre ela e o servidor que guarda a informação. A aplicação deve compreender o formato da informação de volta (a representação), que é geralmente um documento em formato HTML ou XML, onde também pode ser uma imagem ou qualquer outro conteúdo.

REST e SOAP são diferentes, SOAP é um protocolo de mensagem, REST é um modelo de arquitetura de software para sistemas hipermidias distribuídos, na qual a Web é um exemplo deste sistema.

2.11. Infográfico de Comparação entre *SOAP* E *REST*

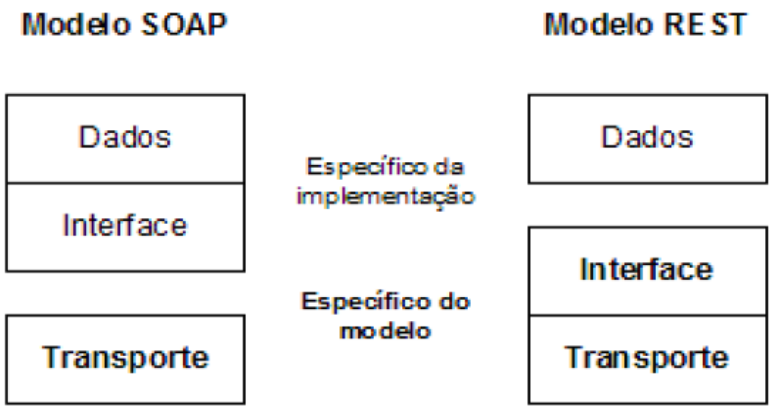


Figura 6. *Organização do modelo SOAP e REST*.
Fonte:[Baker, M[19].*]*

3. DEFINIÇÃO DO MODELO ESCOLHIDO

Com base nas informações analisadas, o modelo de computação distribuída escolhido para o desenvolvimento do trabalho foi o Web Service. Essa escolha teve como base suas características principais como independência de linguagem de desenvolvimento, sistemas operacionais, hardware, plataforma e troca de grandes quantidades de informações por XML. Além disso, o cenário escolhido para o exemplo é mais propício aos serviços web, por se tratar de um ambiente em que são trocadas grandes quantidades de informação e que um servidor é necessário para coordenar toda a parte da regra de negócio da aplicação. Na seção 2.1, os conceitos foram apresentados. De agora em diante, estes conceitos serão discutidos de uma forma mais abrangente.

3.1. Web Service, Discurso Detalhado

Serviços web apresentam um padrão de interface, independentes de plataforma e tecnologia. No cenário atual está expandindo rapidamente devido ao crescimento da necessidade de comunicação e interoperabilidade de aplicações[20].

A organização *world wide web Consortium* (W3C), a qual estabelece os padrões para serviços web define como:

> *Um Web Services é um sistema de software identificado por uma URL onde as interfaces públicas e suas vinculações são definidas e descritas usando XML. Sua definição pode ser descoberta por outros sistemas de software. Estes sistemas podem interagir com serviços de um modo prescrito por sua definição, usando mensagens baseadas em XML, tendo como base a comunicação por protocolos da internet (* Gregory R. Andrews,2000)[21]

Outra definição, mais simplificada para serviços web é

> Um Web Services é uma aplicação de software acessível pela web (ou intranet, redes corporativas) através de uma URL, que é acessada por clientes usando protocolos baseados em XML, como Simple Object Access Protocol (SOAP), enviado através de protocolos aceitos na internet, como HTTP. Clientes acessam uma aplicação baseada em serviços web através de suas interfaces e vinculações, os quais são definidos usando artefatos XML, como o arquivo web Service Definition Language (WSDL)[22].

Web Services se baseiam no conhecimento engajado dos mais maduros ambientes de computação distribuída (como o CORBA e o Java Remote Method Invocation - RMI) permitindo a comunicação direta e a interoperabilidade entre aplicações. Estes oferecem uma forma padronizada para aplicações exporem suas funcionalidades ou se comunicar com outras aplicações através da web (ou uma rede interna), independente da implementação da aplicação, linguagem de programação ou plataforma na qual está sendo executada[23]. Um ponto interessante nas características dos serviços web é que existe a possibilidade de comunicação com serviços externos, ou seja, serviços de terceiros, como exemplo, os serviços do *Google*, com funcionalidades importantes, sendo a mais conhecida o sistema de busca dentro de sites de terceiros.

O motivo mais importante para o aumento do uso de serviços web é o fato de que eles provêm interoperabilidade através de diferentes plataformas, sistemas e linguagens. Além disso, reduzem o custo operacional por permitirem que as organizações a expandam e reutilizem suas funcionalidades de sistemas já existentes. No que diz respeito à arquitetura, um Web Services baseia-se na arquitetura orientada a serviço (SOA), que é um estilo arquitetural que concede a reusabilidade do *software* e o poder de se criar serviços reutilizáveis.

Em um serviço-orientado à arquitetura tem-se:

1. Um serviço implementa uma lógica de negócio e expõe essa lógica através de uma interface bem definida;

2. Um registro onde os serviços publicam suas interfaces a fim de permitirem clientes a descobrir os serviços disponibilizados;

3. clientes (incluindo clientes que podem ser serviços propriamente ditos) que descobrem o serviço usando os registros e acessam os serviços diretamente através de interfaces expostas.

Uma importante vantagem de serviços-orientados à arquitetura é que eles permitem o desenvolvimento de aplicações levemente acopladas que podem ser distribuídas através de uma rede acessível. Para prover esta arquitetura, é necessário:

1. Um mecanismo que permita aos clientes acessarem serviços e registros;

2. um mecanismo que permita diferentes serviços a registrarem sua existência em um registro e um modo para diferentes clientes procurarem no registro um determinado serviço disponível; e
3. um mecanismo para serviços apresentarem uma interface bem definida e um modo para clientes acessarem essas interfaces.

3.2. Benefícios de serviços web

Os serviços web têm ganhado popularidade por causa dos benefícios que os mesmos provêm, estes se distinguem de outros sistemas de software distribuídos, segue abaixo três características:

> **Infraestrutura aberta:** Web Services são implantados que usando protocolos-padrão da indústria e independentes do fornecedor como HTTP e XML, são onipresentes e bem compreendidos. Web Services vão suportar a rede, formatação de dados, segurança e outras infraestruturas já configuradas, o que diminui os custos de entrada e promove interoperabilidade entre os serviços.
>
> **Transparência de linguagem:** Web Services podem interoperar, mesmo se forem escritos em diferentes linguagens de programação. Linguagem como C/C++, Java Perl, Python, Ruby e outras oferecem bibliotecas, utilitários e até frameworks em suporte dos Web Services.
>
> **Design modular:** Web Services são designados para serem modulares em design, para que novos serviços possam ser gerados através da integração e agrupamento de serviços existentes. Imagine, por exemplo, um serviço de monitoramento de estoque integrado com um serviço de pedidos online, para gerar um serviço que automaticamente ordena os produtos apropriados em resposta aos níveis do estoque.[24]

Outros benefícios:

1. Serviços de negócios através da web – pode-se usar um serviço para ampliar suas vantagens através da world wide web (WWW) utilizando-os em suas aplicações;
2. Liberdade de escolha - padrões de serviços web geraram um grande mercado para ferramentas, produtos e tecnologias. Isto dá às corporações uma vasta variedade de escolha, e eles podem escolher as configurações que melhor se encaixem em suas aplicações;
3. Produtividade de programação - serviços web, por criarem um padrão comum de desenvolvimento, ajudam a aumentar a produtividade na programação.

3.3. Evolução de tecnologias e produtos

Serviços web, por causa de sua ênfase em interoperabilidade e independência de plataforma, sistema operacional e linguagem de programação, encontram-se em uma coleção de tecnologias, vários padrões e especificações, muitos dos quais ainda estão sendo definidos e refinados, como exemplo a interoperabilidade e a ordem:

1. Interoperabilidade é um desafio contínuo. Serviços web já têm alcançado um significante degrau de interoperabilidade, mas, futuramente, padrões serão necessários para uma seleção de melhores meio de alcançar tal objetivo.
2. Geralmente, o que acontece para o usuário final que vê um único processamento de negócio é que realmente tudo foi implementado como uma série de estágios em etapas, e cada estágio do processamento pode ser implementado como serviços separados. Todos os serviços devem se coordenar entre si durante os vários estágios do processamento da lógica de negócio a fim de se alcançar o objetivo desejado, por isso padrões são necessários para coordenação de serviços.

3.4. Segurança

A questão de segurança em Web Services cobre um leque enorme de possibilidades, para isto tentaremos subdividir em blocos:

- Segurança de baixo nível; isto é com protocolos básicos que governam comunicações entre um web service, sejam baseados em SOAP ou de estilo REST, e os seus clientes. Geralmente neste nível oferece três serviços:
    1. O cliente e o serviço precisam de segurança a nível de transporte, ou seja que cada um esta se comunicando com o outro sem a interferência de um impostor.
    2. Os dados enviados de um lado para outro precisam ser codificados fortemente o suficiente, para que um interceptor não possa decodifica-los.

3. Cada lado precisa ter certeza que a mensagem recebida é a mesma que foi enviada.

- Autenticação e autorização de usuário o web service oferece aos clientes acesso a recursos. Se um recurso estiver protegido então um cliente precisa de credenciais apropriadas para obter acesso. As credenciais são apresentadas e verificadas através de um processo que geralmente tem duas fases:
    - Primeira: Esta fase é conhecida como autenticação de usuário.
    - Segunda: Consiste em retirar os direitos de acesso do usuário autenticado, selecionando opções para diferentes categorias de acesso separando-os em comum ou administrador disponibilizando maior ou menor nível de acesso.

      Para aprimorar pode-se utilizar o wss (*Web Services security*) ou ws-security que é uma coleção de protocolos que especificam como diferentes níveis de segurança podem ser reforçados em *Web Services* baseados em SOAP. Por exemplo o wss especifica como assinaturas digitais e informação de codificação podem ser inseridas em cabeçalhos SOAP.Lembrando que serviços baseados em SOAP são desenvolvidos para serem de transporte neutro. De acordo WSS é destinado a oferecer abrangente segurança ponto a ponto independente do transporte fundamental.”[2].
    - Questões de segurança para serviços web se preocupam com autenticação, autorização e a certificação da existência de confidencialidade. Padrão de segurança é uma área de alta prioridade para a comunidade de serviços web, pois, à medida que os sistemas evoluem, o grau de segurança acompanha essa evolução. De fato, agora que aplicações são disponibilizadas na web, abrindo mais processamento de negócio e informação, a fim de distribuí-las a clientes, segurança se torna um fator fundamental. Uma das dificuldades é lidar com sistemas de realidades separadas em um ambiente de segurança integrado. Segurança necessita de ser compatível com os mecanismos existentes. Em casos onde clientes precisam acessar

informações protegidas, o mecanismo precisa manter um alto nível de segurança (e a confidência do usuário) e ainda fazê-lo o máximo desobstruído e transparente possível.

3.5. Reusabilidade, Disponibilidade e Escalabilidade

Serviços web são geralmente utilizados em aplicações de larga escala. Com essas aplicações, a disponibilidade, a reusabilidade, e a capacidade de manter a performance e confiabilidade com aumento de carga se tornam fatores importantes a serem considerados. Reusabilidade é o aspecto da possibilidade de utilização do serviço desenvolvido em mais de uma aplicação agilizando e reduzindo tempo com novos desenvolvimentos, buscando atender com qualidade e rapidez uma gama maior de aplicação. Podendo ser reutilizável quanto mais facilmente e automaticamente se pode lidar com a mudança no uso de padrões e configurações do sistema.

Já a disponibilidade representa a probabilidade que um serviço está disponível sempre que desejado. Serviços web que escalam efetivamente podem facilmente lidar com uma grande porção de transações por parte de seus clientes. Da mesma forma que o serviço, a plataforma operacional e as tecnologias empregadas devem gerenciar eficientemente os recursos dos sistemas.

Para conseguir alcançar reusabilidade, disponibilidade e escalabilidade, Web Services devem ser flexíveis o suficiente para serem executados em qualquer configuração de servidor a fim de se antecipar e suportar determinados volumes de clientes, eles devem, também, serem capazes de mudar essas configurações facilmente, quando necessário, sem impactar no que já esta sendo atendido.

3.6. Cenários de serviços web

Quando se desenvolve com a tecnologia *Web Services* deve-se tomar precauções com o grande espectro de cenários, como interações entre padrões de negócio, suplemento da cadeia de gerenciamento, gerenciamento de inventário e até simples serviços (conversores especializados, calculadoras específicas e assim por diante).

Assim, os serviços web, quando usados estrategicamente, podem aumentar significativamente as funcionalidades de uma determinada aplicação, porém não podem ser tomados como soluções apropriadas em todos os pontos de uma aplicação. Enquanto eles fornecem essa riqueza funcional e interoperabilidade, esses serviços podem se tornar de alto custo se levada em conta sua performance.

Devem ser levados em conta alguns itens ao escolher uma solução baseada em serviços web, como requerimentos de interoperabilidade para aplicações em um ambiente heterogêneo, integração para os ambientes que contêm vários sistemas de informação corporativos, tipos de clientes esperados a serem suportados, como tecnologias empregadas e arquitetura da rede comunicação, disponibilidade de ferramentas a fim de implementarem tal solução, nível de sacrifício, em termos de complexidade e performance, que podem ser tolerados a fim de se alcançar as vantagens (interoperabilidade, alcance de diversos clientes e assim por diante) do Web Services.

3.7. Interação com Padrões de Negócio

Serviços web podem ser um meio ideal de se integrar à corporação com múltiplos padrões de negócios, por várias razões: são efetivamente mais baratos do que a integração de informações eletrônicas (EDI), atualmente a solução mais comum disponível para padrões de integração de negócio. EDI requer um significativo investimento em infraestrutura, o que limita a grandes corporações. Uma solução baseada em serviços web se torna plausível para negócios pequenos que não podem investir em uma infraestrutura EDI e podem ser mais eficientes em um ambiente com uma infraestrutura EDI existente. Interações baseadas em serviços fornecem aos padrões, especialmente a maior parte dos padrões com significante tecnologia e investimento em infraestrutura, uma boa vantagem: eles podem usar serviços web para integrar suas aplicações corporativas existentes, baixando os custos e aumentando a interoperabilidade, por exemplo.

3.8. Integrando com Sistemas de Informação Existentes

A maioria das empresas tem feito investimentos no decorrer de muitos anos em tecnologia da informação e recursos de sistema. Geralmente, a maior parte desse investimento é em um sistema de integração de informação corporativo ou sistemas legados. Serviços web podem eficientemente suprir essa necessidade de integrar tecnologias existentes com novas tecnologias. Desde que são baseados no conceito de 'universalmente aceitos', independentes de plataforma ou de padrões, serviços web criam uma camada de integração natural. Assim, é possível fornecer um software que exponha suas funcionalidades em um Web Services, o que torna mais acessível esse tipo de ferramenta ao usuário que necessita acessar certas funcionalidades pela web. Como em nosso cenário a utilização do *Web Services* para gerar carga para vários projetos que antes seria gerados através de integração EDI, necessitando disponibilidade de equipamento para compartilhamento de dados, cargas, e validações localmente realizadas.

3.9. Alcançando diversos clientes

Os serviços web tornam a relação clientes e corporação mais amigável e dinâmica. Isso é possível tornando suas funcionalidades disponíveis, aumentando a experiência de seus clientes, fornecendo serviços como rastreamento de pedido de um cliente, produtos e/ou equipamento a serem separados para envio, identificar localização de produtos e apontar o melhor a ser despachado, e recolhidos.

Clientes podem conseguir essas informações de qualquer lugar e quase sempre usando qualquer aparelho. Alguns clientes podem usar um Web Services em um *browser* (navegador) a fim de acessar estes serviços, outros podem usar dispositivos móveis, como celulares ou *PDA's*.

4. ESTUDO DE CASO

Para realizar estes estudos utilizou-se a necessidade existente em empresas comerciais de integrar sistemas diversos e comunicar módulos desenvolvidos por diversos fornecedores, em que o sistema de ERP utilizado é o Sapiens desenvolvido pela Senior Sistemas, o módulo de gerenciamento de Recursos Humano é o Vetorh desenvolvido pela Senior Sistemas, o módulo de gerenciamento de produção foi desenvolvido internamente e o módulo de intranet foi desenvolvido por uma empresa terceira.

A forma utilizada para levantamento dos que nos levaram a decidir pela utilização de soluções através de Web Service foram:

- Definição prévia dos objetivos a serem atingidos com a integração de informações de áreas distintas, mas que dependiam umas das outras de informações diversas e que muitas vezes geravam: retrabalhos, atrasos, e informações inconsistentes. Sendo a principal meta atender a produção e disponibilizar informações cadastrais a mesma e atender a demanda de informações pessoais aos funcionários.

- Entrevistas: Foi realizado reuniões com os integrantes das áreas principais a serem atendidas, verificando as informações que cada área necessitava e o que efetivamente poderia ser disponibilizado, atendendo a legislação trabalhista e as necessidades da empresa.

- Coleta de dados: Após as reuniões foi coletado individualmente, as principais ferramentas de controle utilizadas na empresa e as que mais retrabalhos e problemas apresentavam, devido a falta de integração de informações, bancos de dados com informações duplicadas, planilhas eletrônicas, documentos, relatórios do sistema e apontamentos da produção.

- Formalização e definição de datas para apresentação das soluções encontradas, testes, validação e homologação.

Apresentamos assim o cenário para estudo:

- Sapiens: Desenvolvido em Delphi sem nenhuma interface WEB.

- Vetorh: Desenvolvido em Delphi sem nenhuma interface WEB.

- Controle de Produção: Desenvolvido em PHP.

- Intranet: Desenvolvido em .NET com ferramentas da Microsoft SharePoint .

Cenário 1.

Tem-se o sistema de ERP (Sapiens) com as políticas de segurança todas consolidadas, grupos criados, usuários cadastrados e controle efetivo “funcionando” e um módulo de produção carente de controle de segurança, sem autenticação efetiva, vazamento de informações e acessos sem consolidação.

Cenário 2.

Necessidade de disponibilizar, no módulo de intranet que já estava se tornando extranet, informações pertinentes a recursos humanos, como quadro de vagas, dados de colaboradores, necessidade de departamentos, solicitação de pessoal.

Soluções apresentadas para o cenário 1:

Expansão do espaço em disco do servidor para replicar as informações de colaboradores do sistema de recursos humanos com os dados de colaboradores disponíveis por departamento e unidade, com a integração dos acessos pertinentes no sistema de ERP para que se pudesse consolidar usuários ativos efetivamente na empresa (Trabalhando), com o que está no cadastro no sistema de segurança do ERP, gerando assim informações para serem consolidadas no acesso ao módulo de produção. Estes dados seriam atualizados diariamente por um *dblink* ou processo EDI (Electronic Data Interchange - significa troca estruturada de dados através de uma rede de dados qualquer)[25] que seria ativado em uma determinada hora, pois os bancos utilizado pelos sistemas eram diferentes (Oracle e SqlServer), e estes dados gerados ficariam no banco do sistema de produção SqlServer, gerando dados replicados e armazenamento inútil bem como serviços de manutenção acompanhamento de processos, monitoramento de processo, etc. Esta integração deveria ser realizada pelas empresas responsáveis pelos ERP Sapiens e Rh – Vetorh. Ou adquirimos um portal de Rh ao qual possuem páginas de consultas que deveriam ser lidas pelo módulo de produção para se realizar a liberação de acesso e apresentar as informações do operador autenticado (*logado*).

Cenario 2, Soluções apresentadas para o cenário 2:

Desenvolvimento pela empresa de suporte do sistema de Recursos Humanos de módulo de exportação de informações do sistema para gerar as informações que seriam integradas ao portal desenvolvido com SharePoint. Esta exportação seria via EDI, para gerar estes dados, causando demora de apresentação no site, falha na exportação, manutenção constante,bem como o desenvolvimento no outro lado para carregar esta informações, monitoramento de processos e carga, etc.

4.1. Descrição da Aplicação Exemplo

Um cenário característico de computação distribuída é o de necessidade de integração e compartilhamento de informações entre sistemas e/ou departamentos distintos, é o cenário apresentado, em que após varias reuniões e propostas com orçamentos abusivos, chegou-se a um momento de quebra de paradigma com a implantação de soluções que trariam liberdade sem grandes investimentos ou compras de ferramentas novas ou desenvolvimentos complexos, investimento em infra-estrutura e servidores grandiosos, para se chegar as informações de forma homogênea sem replicação e com segurança.

Após várias reuniões e orçamentos chegou-se a hipóteses do se utilizar a opção de web services, para se atender algumas situações, sem abrir acessos aos bancos de dados a terceiros ou pessoas com pouca experiência, mas devido ao pouco conhecimento sobre a tecnologia de web services e dúvidas da eficiência, adotou-se a solução como uma alternativa, enquanto os desenvolvedores melhoravam o escopo de projeto e os valores para a execução de outra maneira.

Solução proposta, com *Web Services*:

Necessita-se de um servidor configurado com o Glassfish, e a princípio três (03) *Web Services* que retornariam:

1. Dados de colaboradores: Informação oriunda do Vetorh (Recursos Humanos).
2. Dados de Permissão: Informação oriunda do Sapiens (ERP).

3. Dados do Quadro de Vagas: Informação oriunda do Vetorh (Recursos Humanos).

A decisão de se utilizar o Glassfish foi tomada mediante comparações de onde poderíamos chegar com um servidor de aplicação , este deveria incluir uma pilha de serviços Web (Metro), EJBs, Java Persistence, JMS, JavaMail , etc, bem como, através de levantamento com o fornecedor da solução do RH, Senior Sistemas desenvolvedora do Vetorh, onde este informou que o Sistema Vetorh estava evoluindo, e em pouco tempo necessitaríamos cria um ambiente que desse suporte a tecnologia Java, e que estas novas soluções estariam sendo desenvolvidas com Glassfish, acabando assim com nossas duvidas, pois mesmo que não funcionasse a contento nossa solução interna a estrutura poderia ser utilizada para outra demanda, não ficando perdido o investimento.

Pré-requisito: Este servidor não poderia parar e estar disponível a qualquer momento.

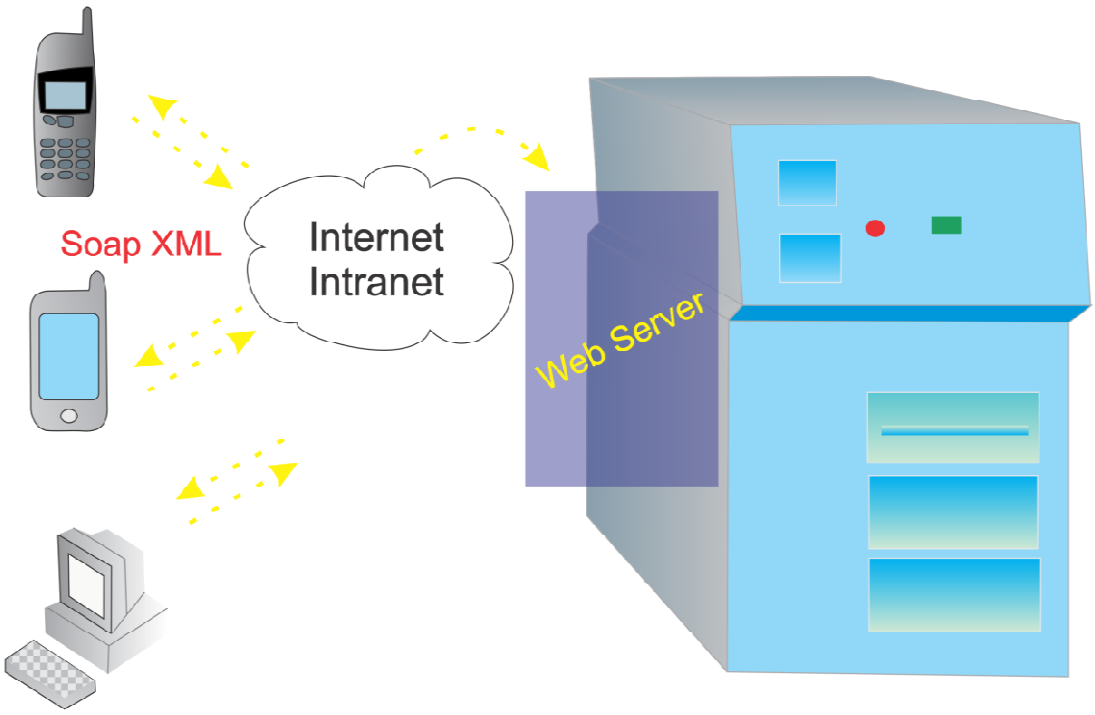


Figura 7. Demonstração da aplicação.

4.2. Diagrama de Caso de Uso

O diagrama de caso de uso descreve a funcionalidade proposta para um novo sistema, que será projetado.

> Documento narrativo que descreve a sequência de eventos de um ator que usa um sistema para completar um processo.
>
> Ivar Jacobson

Um caso de uso representa uma unidade discreta da interação entre um usuário (humano ou máquina) e o sistema. Um caso de uso é uma unidade de um trabalho significante. Por exemplo: o "*login* para o sistema", "registrar no sistema" e "criar pedidos" são todos casos de uso. Cada caso de uso tem uma descrição da funcionalidade que irá ser construída no sistema proposto. Um caso de uso pode "usar" outra funcionalidade de caso de uso ou "estender" outro caso de uso com seu próprio comportamento.

Casos de uso são tipicamente relacionados a "atores". Um ator é um humano ou entidade (máquina) que interage com o sistema para executar um significante trabalho[26].

A solução desenvolvida independe de plataforma de cliente ou equipamento, ela simplesmente disponibiliza as informações mediante as requisições realizadas e podem ser atualizadas, renovadas e atualizadas a qualquer momento sempre tomando o cuidado de não interromper os clientes antigos. Podem existir várias requisições

UseCase Diagram

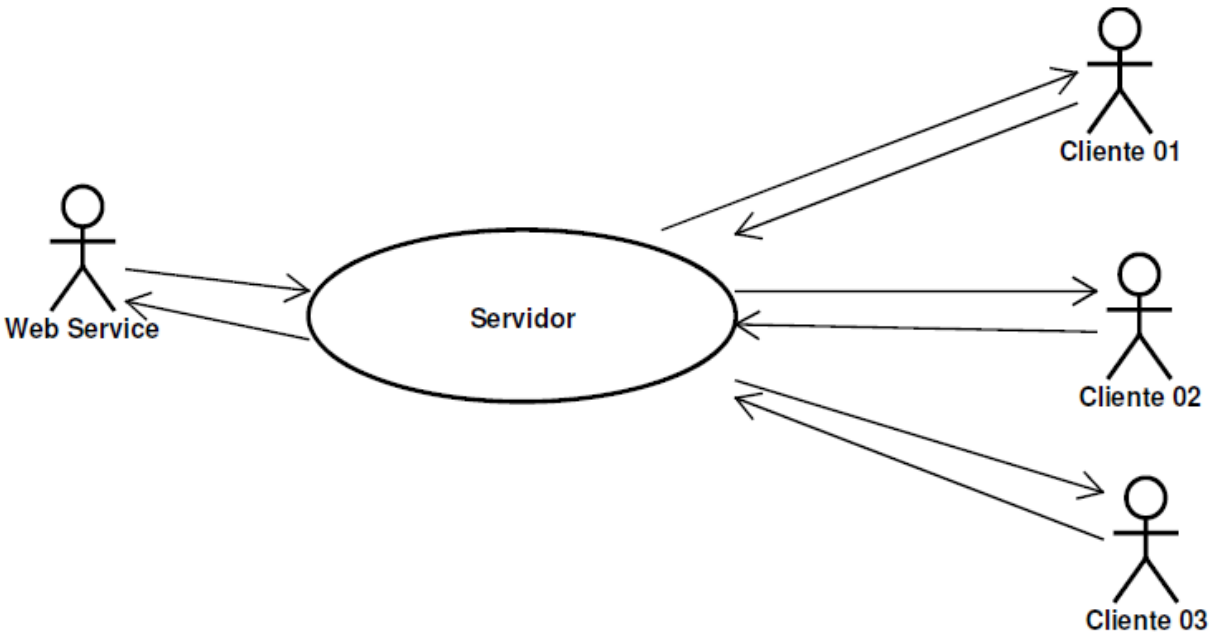


Figura 8. *Diagrama de caso de uso do web service de retorno de dados cadastrais* UML

Web Service: um unico web service pode atender a varios clientes de departamento diferentes, ex: O Web service que retorna informações cadastrais, pode atender o sistema de segurança, produção e rh.

4.3. Especificação do Projeto

Fornecer informações cadastrais dos sistemas de gerenciamento do recursos humanos para os módulos de segurança do sistema de produção, web, sistemas estes desenvolvidos por outras equipes da empresa e por prestadores de serviços externos, não sendo necessário a abertura das informações acesso de usuário máster e senha dos bancos de dados, gerando web service para atender a necessidade de informações dos sistemas citados.

4.4. Tecnologia

Para que a aplicação apresentada pudesse ser concreta e para que o objetivo deste trabalho fosse alcançado, foi utilizado:

1. Informações de Colaboradores: Sistema VetorH (Módulo Rubi), desenvolvido pela Senior Sistemas.

2. Informações de Vagas: Sistema VetorH (Módulo Recrutamento e Seleção), desenvolvida pela Senior Sistemas.
3. Posteriormente já foi solicitado o desenvolvimento para a área comercial e pedidos, mas até o momento não foi customizado.
4. O Banco de dados utilizado será o Oracle 11g, O servidor foi configurado com o GlassFish versão V3, Netbeans 7.0, JDK 1.7.

4.5. Arquitetura

O Diagrama de pacotes, ou diagrama de módulos, definido pela UML descreve os pacotes ou pedaços do sistema divididos em agrupamentos lógicos mostrando as dependências entre estes, ou seja, pacotes podem depender de outros pacotes.A arquitetura do Web Services é representada através do diagrama de pacotes. Tal diagrama contém os pacotes usados no exemplo, bem como acontece a comunicação entre eles.

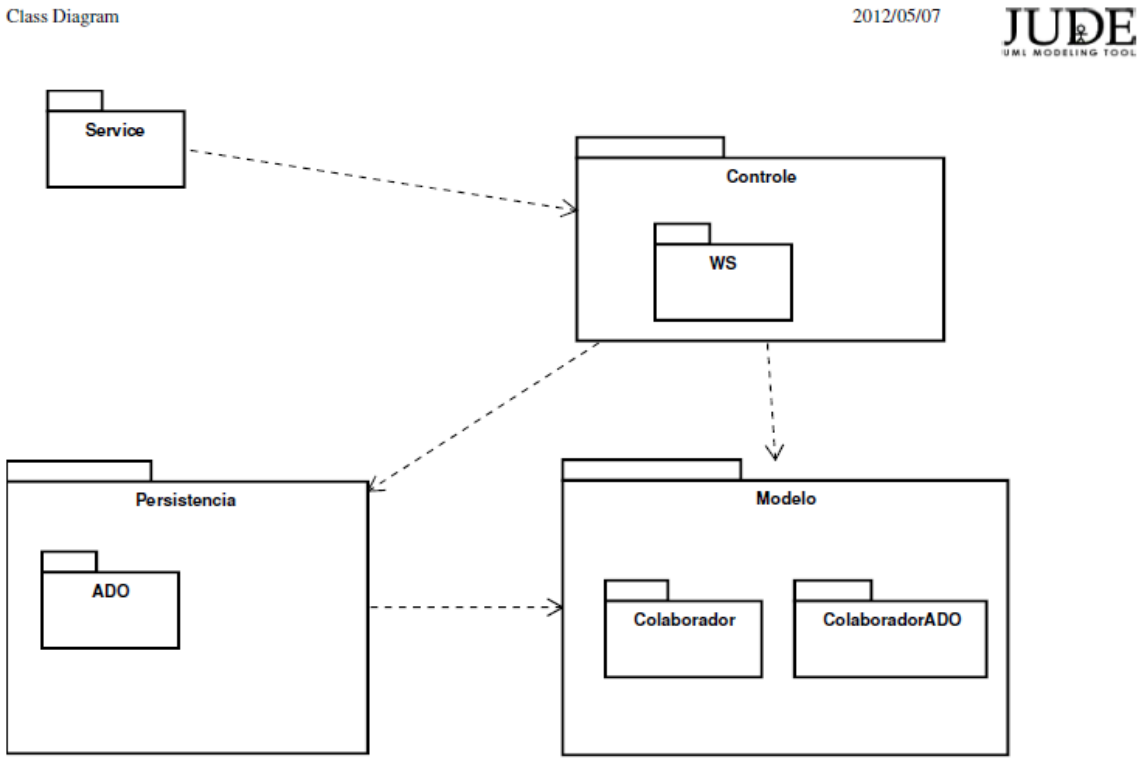


Figura 9. *Diagrama de caso de uso do web service de retorno de dados cadastrais*.

4.6. Diagrama de implantação

O diagrama de implantação modela o *hardware* do ambiente de implantação. Cada nó em um diagrama de implantação normalmente representa um tipo de *hardware*. Um nó também pode representar um ser

humano ou uma unidade organizacional, ou, mais precisamente, a função que uma pessoa pode realizar[27].

O ambiente físico teórico da aplicação exemplo está descrito na figura 10:

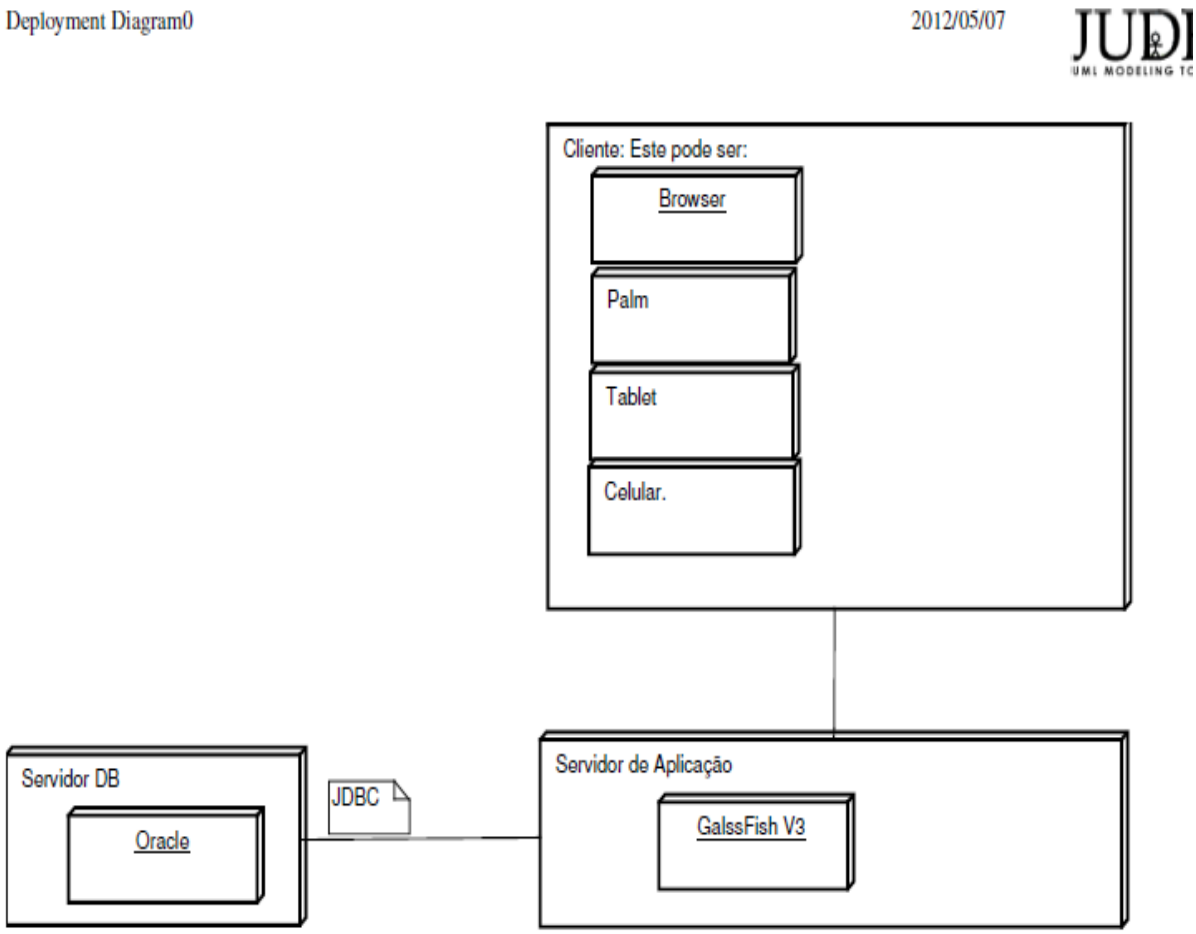


Figura 10. *Diagrama de implantação*.

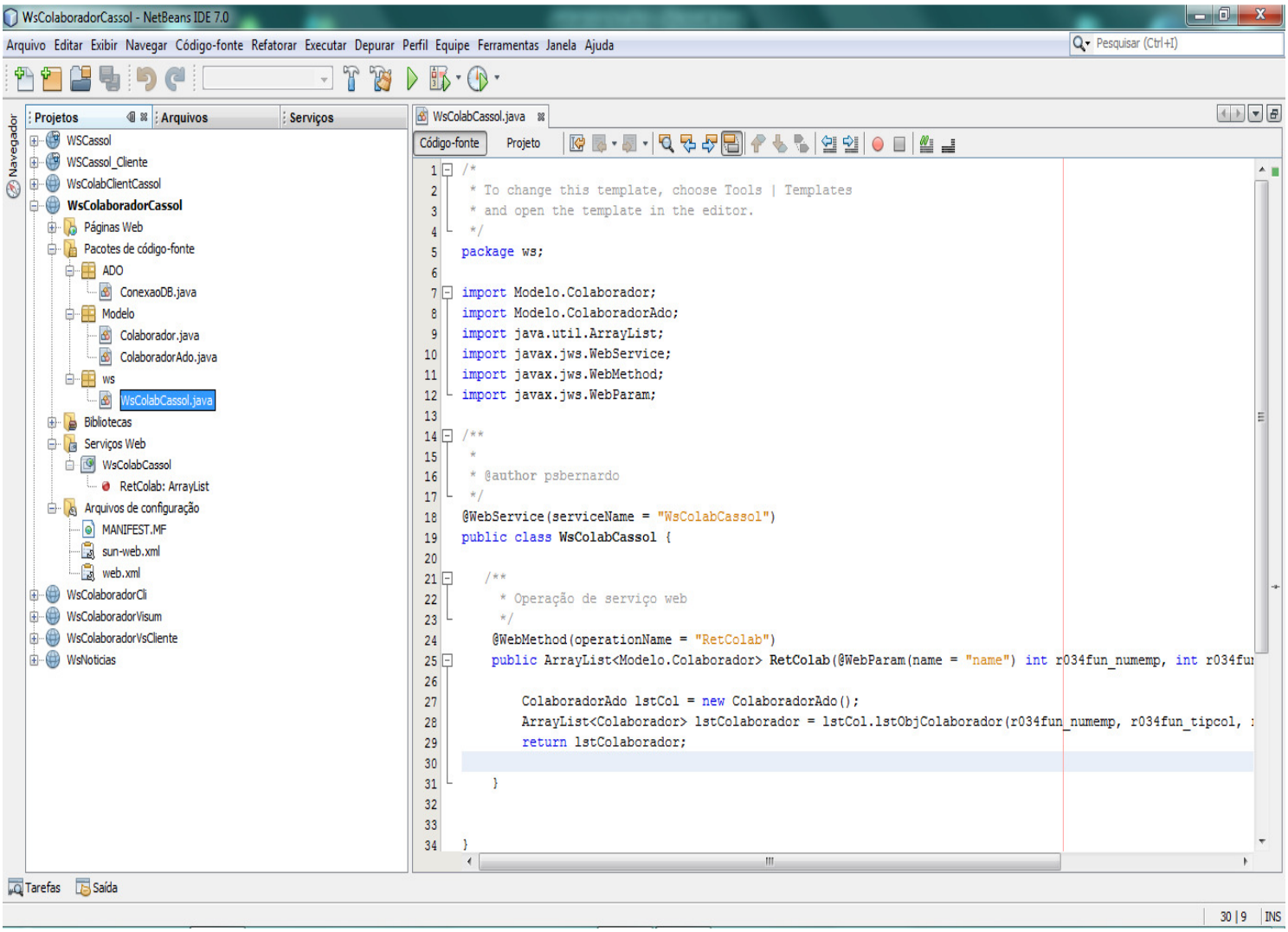
4.7. Construindo e Consumindo um Web Service

Nesta seção será apresentada a maneira como o Web Service foi construído, publicado e consumido, com exemplos práticos e demonstrações de fragmentos de códigos Java e um simulador de como ele poderia ser utilizado em uma página web pois por motivos de proteção de direitos autorais não poderemos expor o código utilizado no desenvolvimento da aplicação original.

4.8. Implementando um Web Service

A primeira tarefa a executar para se implementar um Web Service é criar uma classe que detenha todo seu escopo, e nessa aplicação exemplo foi criada uma classe chamada WSColabCassol. Esta classe fará uso de toda a estrutura da aplicação já desenvolvida, porém provê a funcionalidade de transferir os dados de colaboradores de uma forma mais simples.

O código-fonte da classe WSColabCassol é mostrado a seguir:




```
1  /*
2   * To change this template, choose Tools | Templates
3   * and open the template in the editor.
4   */
5  package ws;
6
7  import Modelo.Colaborador;
8  import Modelo.ColaboradorAdo;
9  import java.util.ArrayList;
10 import javax.jws.WebService;
11 import javax.jws.WebMethod;
12 import javax.jws.WebParam;
13
14 /**
15  *
16  * @author psbernardo
17  */
18 @WebService(serviceName = "WsColabCassol")
19 public class WsColabCassol {
20
21     /**
22      * Operação de serviço web
23      */
24     @WebMethod(operationName = "RetColab")
25     public ArrayList<Modelo.Colaborador> RetColab(@WebParam(name = "name") int r034fun_numemp, int r034fu
26
27         ColaboradorAdo lstCol = new ColaboradorAdo();
28         ArrayList<Colaborador> lstColaborador = lstCol.lstObjColaborador(r034fun_numemp, r034fun_tipcol,
29         return lstColaborador;
30
31     }
32
33
34 }
```

Figura 11. Codigo do exemplo da aplicação que consome o web service por código java.

O trecho entre a linha 7 e a linha 12, mostra a qual pacote pertence a classe e as devidas importações necessárias para a classe, sendo esta a parte básica do serviço.

A anotação na linha 18 informa que esta classe em questão será um Web Services.

Esse é o motivo da anotação @WebService*()* aparecer antes da declaração da classe. Após a classe ser informada que ela será um serviço, deve-se desenvolver o método (função que realizará a transferência) que será provido, feitos assim como na linha 24. Esta anotação informa que o método descrito ali será disponibilizado pelo Web Services. Uma vez que a classe e os métodos foram declarados, os atributos do método seguem o mesmo princípio assim como mostrado no código entre as linhas 27 e 29 onde é carregado a classe Colaborador e retornado ao cliente os dados solicitados.

Agora, falta apenas registrar o serviço e descrevê-lo no arquivo que fará referência ao serviço disponibilizado, o (Web Service Definition Language) WSDL.

Segue abaixo, o arquivo WSDL da aplicação apresentada:

Antes de inicar a descrição do código, devemos esclarecer os elementos básicos do WSDL:

WSDL é documento XML que descreve os elementos dos serviços web:

<types>:descrevem os de dados suportados pelo serviço.

<message>: especifica os padrões de entrada e saída de dados dos *web services.*

<portType>: descreve os agrupamentos lógicos das operações. São as operações executadas pelo *web service.*

<binding>: apresentados os protocolos de comunicação que os *web services* utilizam.

<operation>: permite a especificação das assinaturas dos métodos disponibilizados.

<definitions>: elemento padrão de todos os documentos WSDL, permite efetuar descrições sobre *schemas* e *namespaces.*

Namespaces: conceitos de reaproveitamento e organização de códigos são implementados também. A seguir, apresentamos os principais *namespaces* já disponíveis para WSDL.

http://schemas.xmlsoap.org/wsdl: *namespace* de WSDL.

http://schemas.xmlsoap.org/wsdl/soap: *namespace* que permite a utilização de WSDL com SOAP

http://schemas.xmlsoap.org/wsdl/http: *namespace* que permite HTTP, GET e POST

http://schemas.xmlsoap.org/wsdl/mime: *namespace* que permite a vinculação de MIME aos WSDLs

http://schemas.xmlsoap.org/soap/encoding: *namespace* que permite a codificação segundo padrão SOAP

http://schemas.xmlsoap.org/soap/envelope: *namespace* que permite a codificação segundo padrão SOAP 1.1

http://www.w3.org/2001/XMLSchema-instance: *namespace* de instância definido pelo padrão XML

http://www.w3.org/2001/XMLShema: *namespace* de *schemas* definido conforme regras do XML[28]

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <!-- Published by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.4-b01-. -->
3  <!-- Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.1.4-b01-. -->
4  <!--
5   Generated by JAX-WS RI at http://jax-ws.dev.java.net. RI's version is JAX-WS RI 2.2.1-hudson-28-.
6  -->
7  definitions targetNamespace="http://ws" name="WsColabCassol">
8  -<types>
9   -<xsd:schema>
10     <xsd:import namespace="http://ws" schemaLocation="http://localhost:8080/WsColaboradorCassol/WsColabCassol?xsd=1"/>
11    </xsd:schema>
12  </types>
13 -<message name="RetColab">
14    <part name="parameters" element="tns:RetColab"/>
15  </message>
16 -<message name="RetColabResponse">
17    <part name="parameters" element="tns:RetColabResponse"/>
18  </message>
19 -<portType name="WsColabCassol">
20  -<operation name="RetColab">
21     <input wsam:Action="http://ws/WsColabCassol/RetColabRequest" message="tns:RetColab"/>
22     <output wsam:Action="http://ws/WsColabCassol/RetColabResponse" message="tns:RetColabResponse"/>
23    </operation>
24  </portType>
25 -<binding name="WsColabCassolPortBinding" type="tns:WsColabCassol">
26    <soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
27  -<operation name="RetColab">
28     <soap:operation soapAction=""/>
29    -<input>
30       <soap:body use="literal"/>
31     </input>
32    -<output>
33       <soap:body use="literal"/>
34     </output>
35    </operation>
36  </binding>
37 -<service name="WsColabCassol">
38  -<port name="WsColabCassolPort" binding="tns:WsColabCassolPortBinding">
39     <soap:address location="http://localhost:8080/WsColaboradorCassol/WsColabCassol"/>
40    </port>
41  </service>
42  definitions>
```

Figura 12. Codigo do exemplo do XML gerado pela aplicação que utiliza a chamada do web service por código java.

Na linha 1 encontra-se a versão do arquivo e a formatação empregada no uso dos caracteres que irão compor o serviço. A linha 7 possui a tag <definition>. Esta tag inicia a definição do schema a ser usado pelo Web Services. No schema estão descritas informações como os tipos dos atributos esperados pelo(s) métodos(s) oferecidos pelo(s) serviço(s). A tag também define a localização do serviço, definindo para o mesmo um namespace e define o envelope SOAP do Web Services.Na linha 8, é associado ao namespace o seu devido schema. A partir da linha 14 até a linha 20 são

definidos os nomes das mensagens que serão trocadas entre o provedor do serviço e a aplicação que o consome. "RetColab" diz respeito à mensagem do lado cliente e "RetColabResponse" diz respeito à mensagem que virá do servidor.

A partir da linha 21, inicia-se a identificação do portType dos serviços, que no caso da aplicação apresentada será somente um único serviço. A porta será responsável por interpretar a mensagem que vem do cliente e a mensagem que é enviada do servidor.

Os nomes que foram definidos entre a linha 14 e a linha 20 está sendo referido na tag pertinente a porta. A partir da linha 27 é explicitado o arquivo wsdl quais serão as características do envelope SOAP a serem trocadas entre provedor de serviço e consumidor deste. A linha que inicia essas explicitações diz qual porta será usado para qual serviço e, logo abaixo, na linha 28, é dito qual será o tipo de informação que será trocada e sob qual protocolo este irá ser transportado e, no caso da aplicação apresentada, o tipo será de documento (document) e o protocolo será o HTTP.

A próxima etapa do arquivo, que está na linha 29, define qual é o nome do operação que ele irá se referenciar quando uma requisição chegar a ele.

E até a linha 37 é descrito que o transporte dos atributos passados como parâmetro à operação esperada dar-se-á na forma literal que é a forma que os atributos não sofrerão nenhum tipo de criptografia antes e nem depois de serem enviados.

Finalmente, a linha 41 explicita o nome do serviço a ser procurado no provedor do mesmo, e a linha 42 mostra qual a sua porta e quem será responsável por sua ligação com o serviço que está no endereço fornecido na linha 43. E, assim, foi definido o arquivo wsdl para o serviço apresentado neste trabalho.

4.9. Consumindo um Web Services

Agora que o Web Services já está disponível e pronto para aceitar requisições, cria-se uma aplicação web que faça o consumo do Web Services. Essa aplicação cliente não fará uso de nenhuma biblioteca. Sendo a aplicação

cliente portável, seria injustificável ter que obrigar uma determinada aplicação que deseje fazer uso de um serviço, ter que fazer uso de alguma biblioteca para tal.

Primeiro a se fazer será construir o schema que será usado pelo arquivo wsdl do cliente. Pela imagem se demonstra o consumo do web service desenvolvido para atender as necessidades do RH sobre dados de funcionarios, na Figura 13,na linha 22 identificamos o consumo do web service que ira gerar uma lista com nomes de funcionários através de uma aplicação em texto desenvolvida em Java mas poderia ser qualquer outra linguagem.

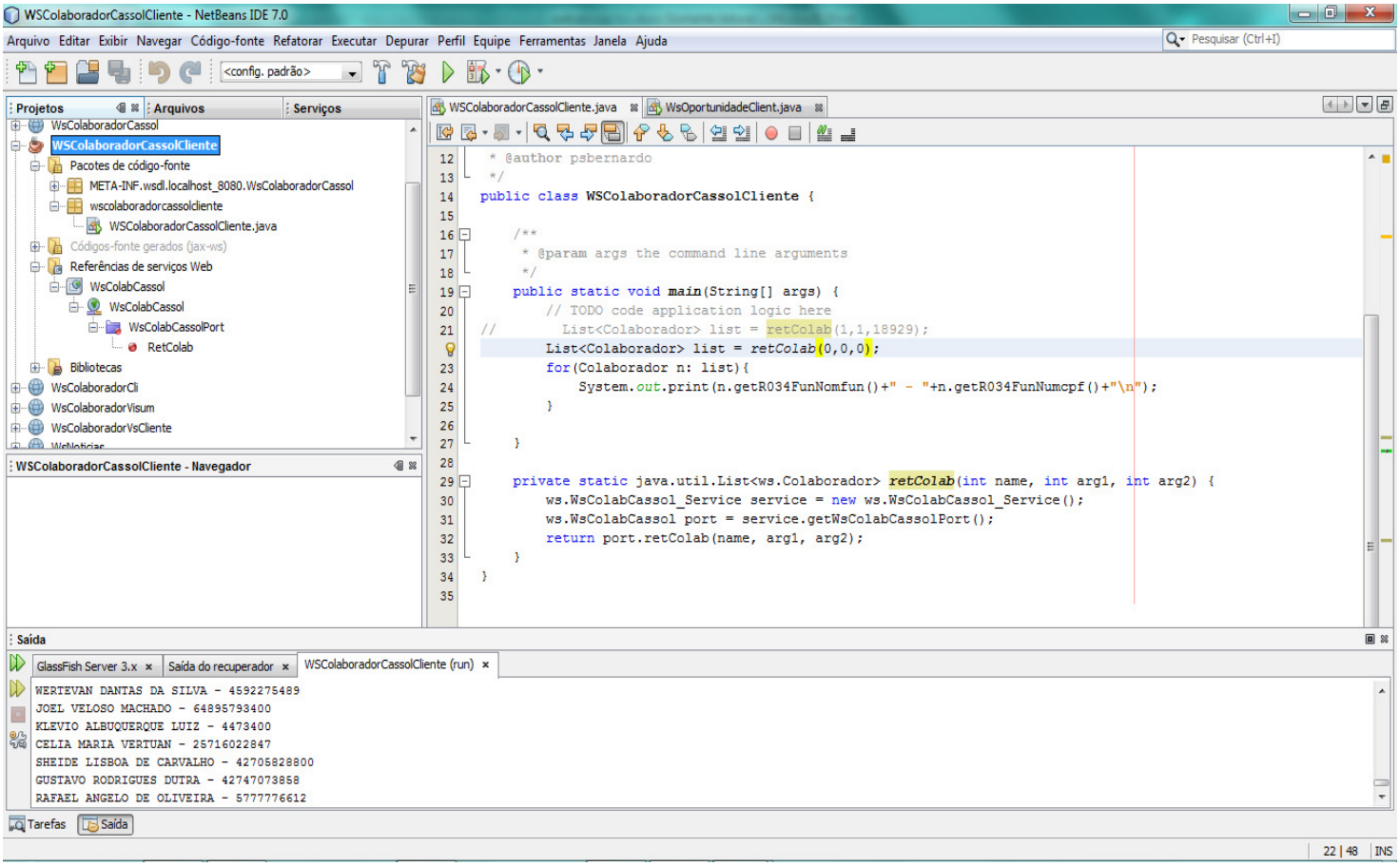


Figura 13. Codigo do exemplo da aplicação que consome web service por código Java, jse.

O web service pode ser utilizado por uma aplicação simples através de linha de comando,,como é demonstrado na Figura 14, o web service é consumido através de uma aplicação que é executada diretamente no prompt do DOS:

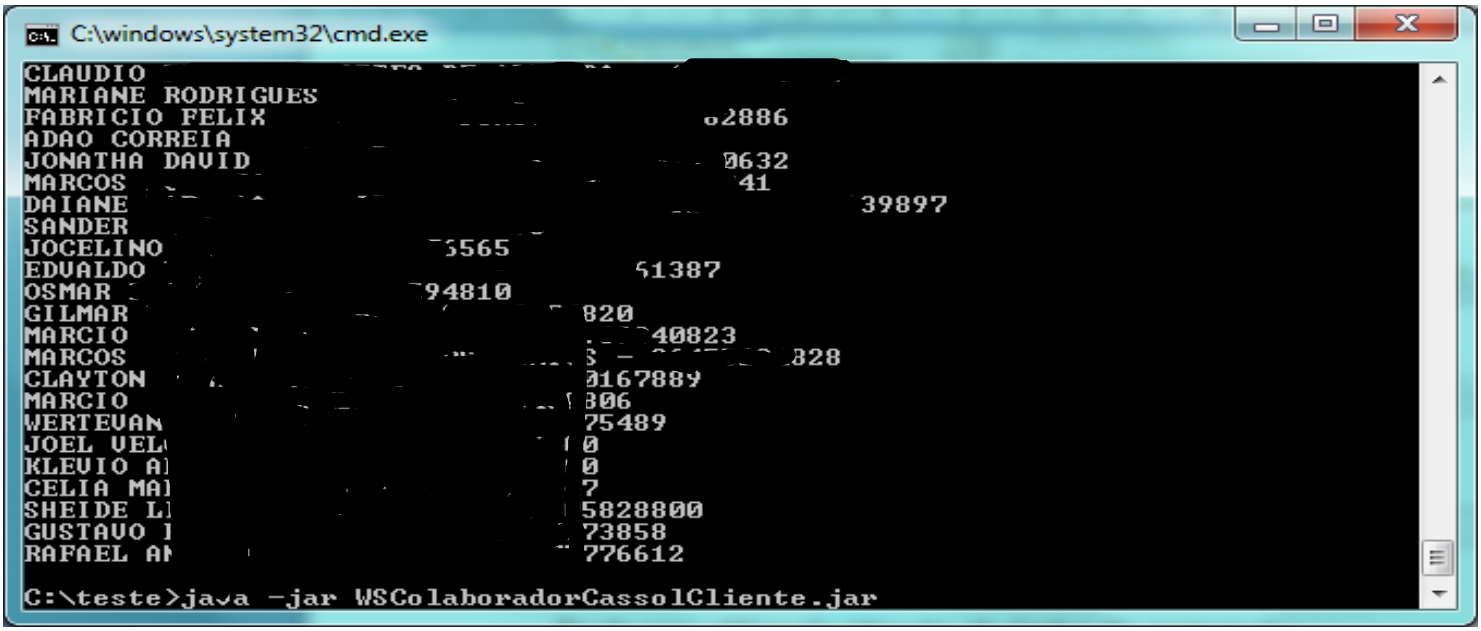


Figura 14. Codigo do exemplo da aplicação que consome web service por código Java, jse, direto no prompt do sistema operacional Windows.

O mesmno web service pode ser consumido por uma aplicação Web, como é apresentato abaixo na Figura 15:

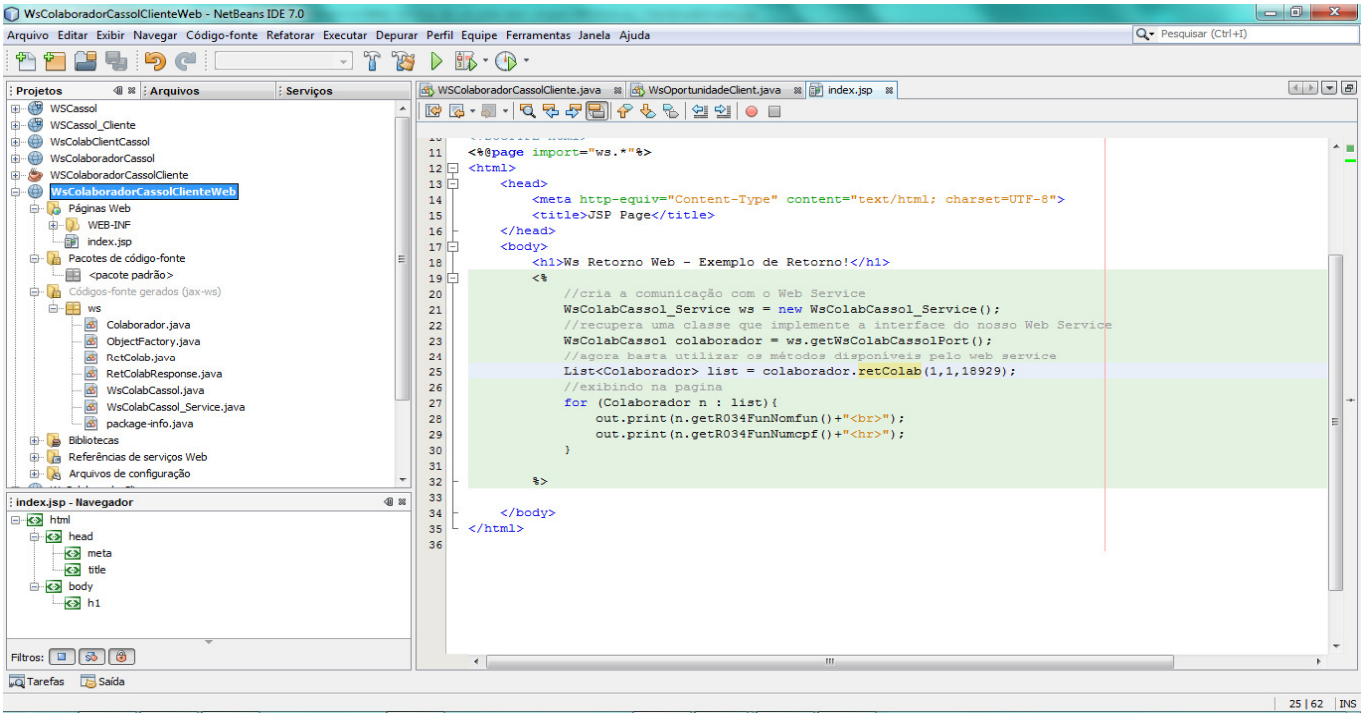


Figura 15. Codigo do exemplo da aplicação que consome web service por código Java, jee.

Chamada do aplicativo que consome o web server na web através do browser:

http://localhost:8080/WsColaboradorCassolClienteWeb/

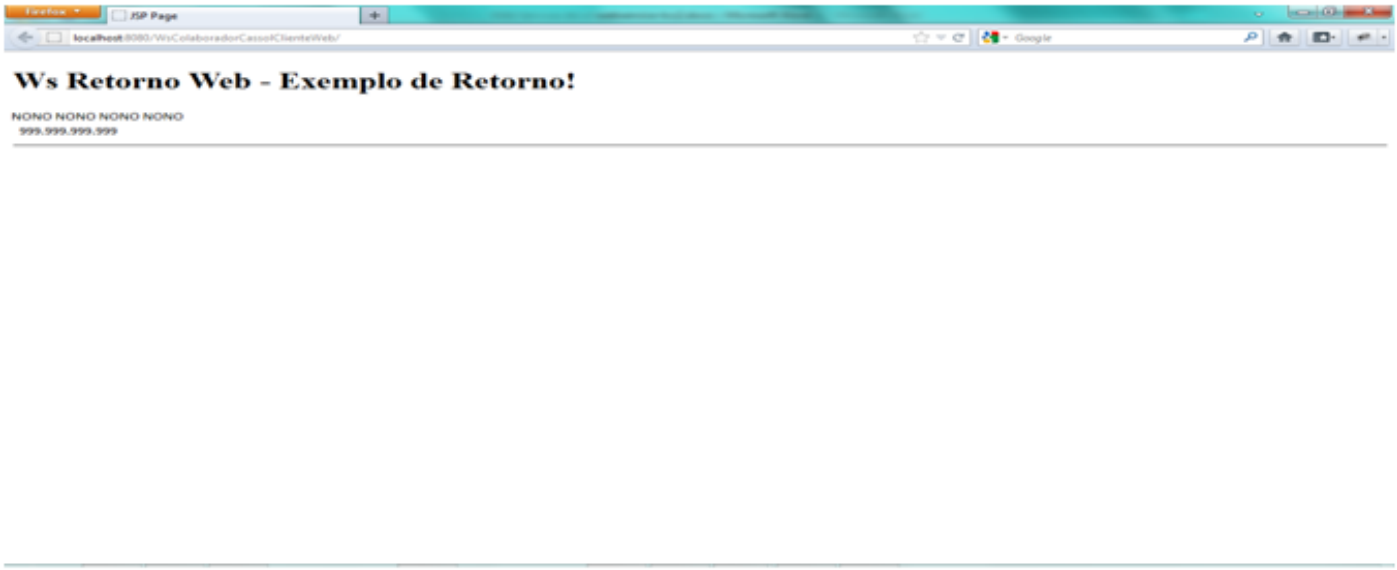


Figura 16. Imagem do retorno da chamada web do consumo de web service.

5. CONCLUSÃO

Focou-se nos modelos baseados em SOAP para atingir o objetivo que é apresentar soluções com baixo custo utilizando as inovações tecnológicas para atender e complementar as soluções da área de TI. Entre as situações apresentadas foram levantadas vantagens, desvantagens e características marcantes de cada um. Foi utilizado o SOAP pelas características da solução pretendida, a facilidade de documentação e pelo domínio de conhecimento dos profissionais envolvidos.

Dessa forma, foi possível constatar que cada modelo possui peculiaridades que fazem com que se adequem a determinado tipo de cenário de aplicação.

Com base no estudo feito e no cenário do estudo de caso, foi escolhido um modelo específico (*web service*) para o desenvolvimento de uma aplicação. Além disso, o volume de pesquisas realizadas na área de serviços web evidencia a sua importância e o grande interesse ainda remanescente por parte de organizações e instituições acadêmicas, por essas razões, tal modelo foi escolhido para ser utilizado.

No estudo de caso desenvolvido, foi exemplificado de modo prático um Web Service, desde a sua fase inicial, passando pela implementação até sua publicação para consumo de terceiros. Mostrando também a interoperabilidade de formas diferentes de acesso, justificando assim um dos objetivos do trabalho.

Para auxiliar no desenvolvimento da aplicação, frameworks como *Java*-ServerFaces e a IDE NetBeans foram utilizados, trazendo desta forma facilidades que antes não existiam, tornando mais fácil o aprendizado e desenvolvimento do Web Service e das aplicações clientes.

Em um contexto geral, com a existência de sistemas legados, possibilidades de consumo de serviços de terceiros e levando em consideração as características apresentadas ao longo do trabalho, o conceito mais importante foi o de interoperabilidade, pois, permitiu-se que uma aplicação construída em uma linguagem diferente consumisse o serviço sem problemas, caracterizando assim a integração entre sistemas de plataformas e linguagens diferentes.

Por fim, ao longo desse trabalho foram observados alguns tópicos que não faziam parte do escopo inicial, mas foram entendidos como interessantes fontes de trabalhos futuros: propõe-se a pesquisa mais detalhada em soluções para o gerenciamento de serviços por meio de outros serviços web, mostrando quais são os pontos fracos e fortes desse modelo de desenvolvimento de serviços web; Além disso, evoluir o estudo de caso de forma a agregar aspectos de segurança, confiabilidade, tolerância a falhas e disponibilidade é outro ponto interessante, visto como trabalho futuro.

6. REFERENCIA

[1] Deitel, Choffnes. *Sistemas Operacionais*. Prenticel-Hall. 2007.

[2] Kalin, Martin. Java Web Services: Implementando. Alta Books. 2010.

[3] Wong, W., Kane, M., Ricciuti, M.: The new buzz - "Web Services"try to rise above high-tech din CNET News.Com. (2000) http://news.com.com/2009-1017-275484.html?legacy=cnet&tag=tp pr [02-12-2004]

[4] Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H. F., Thatte, S., Winer, D.: Simple Object Access Protocol (SOAP) 1.1. W3C Note, World Wide Web Consortium. (2000) http://www.w3.org/TR/SOAP [02-12-2004]

[5] Kalin, Martin. Java Web Services: Implementando. Alta Books. 2010.

[6] Kalin, Martin. Java Web Services: Implementando. Alta Books. 2010.

[7] Box, D., Ehnebuske, D., Kakivaya, G., Layman, A., Mendelsohn, N., Nielsen, H. F., Thatte, S., Winer, D.: Simple Object Access Protocol (SOAP) 1.1. W3C Note, World Wide Web Consortium. (2000) http://www.w3.org/TR/SOAP [02-12-2004]

[8] Deitel, Choffnes. *Sistemas Operacionais*. Prenticel-Hall. 2007.

[9] Kalin, Martin. Java Web Services: Implementando. Alta Books. 2010.

[10] Gudgin, M., Hadley, M., Mendelsohn, N., Moreau, J., Nielsen, H. F.: SOAP Version 1.2 Part 1: Messaging Framework W3C Recommendation, World Wide Web Consortium. (2003) http://www.w3.org/TR/soap12-part1/ [02-12-2004]

[11] Deitel, Choffnes. *Sistemas Operacionais*. Prenticel-Hall. 2007.

[12] Deitel, Choffnes. *Sistemas Operacionais*. Prenticel-Hall. 2007.

[13] Kalin, Martin. Java Web Services: Implementando. Alta Books. 2010.

[14] Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., Orchard, D.:Web Services Architecture W3CWorking Group Note, WorldWideWeb Consortium. (2004) http://www.w3.org/TR/ws-arch/ [24-11-2004]

[15] Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., Orchard, D.:Web Services Architecture W3CWorking Group Note, WorldWideWeb Consortium. (2004) http://www.w3.org/TR/ws-arch/ [24-11-2004]

[16] Fielding, R. T.: Architectural styles and the design of networked-based software architectures Dissertaçãao de Doutoramento Dept. of Information and Computer Science, University of California, Irvine (2000)

[17] Berners-Lee, T., Fielding, R. T., Masinter, L.: Uniform Resource Identifiers (URI): Generic Syntax RFC 2396, Internet Engineering Task Force. (1998) http://www.ietf.org/rfc/rfc2396.txt [28-08-2012]

[18] Fielding, R. T.: Architectural styles and the design of networked-based software architectures Dissertaçãao de Doutoramento Dept. of Information and Computer Science, University of California, Irvine (2000)
[19] Baker, M.: Protocol independence. (2004) http://www.markbaker.ca/2002/09/Blog/2004/10/29#2004-10-protocols [08-05-2012]
[20] Mauricio Reckziegel**,** Entendendo os WebServices**,**sexta-feira, 23/06/2006 às 12h06**,** http://imasters.com.br/artigo/4245/web-services/entendendo-os-webservices.
[20] Baker, M.: Protocol independence. (2004) http://www.markbaker.ca/2002/09/Blog/2004/10/29#2004-10-protocols [08-05-2012]
[20] Gregory R. Andrews. Foundation of multithreaded, parallel, and Distributed Pogramming 1st Edition. Ed. Addison-Wesley. 2000**.**

[21] Baker, M.: Protocol independence. (2004) http://www.markbaker.ca/2002/09/Blog/2004/10/29#2004-10-protocols [08-05-2012]

[22] Gregory R. Andrews. Foundation of multithreaded, parallel, and Distributed Pogramming 1st Edition. Ed. Addison-Wesley. 2000
[23] Gregory R. Andrews. Foundation of multithreaded, parallel, and Distributed Pogramming 1st Edition. Ed. Addison-Wesley. 2000
[24] Kalin, Martin. Java Web Services: Implementando. Alta Books. 2010
[25] http://pt.wikipedia.org/wiki/EDI,acessado em 30/08/2012.
[26] Wikkipedia. *Diagrama de Caso de Uso*. 2012. URL: http://pt.wikipedia.org/wiki/Diagrama_de_caso_de_uso. Acessado em 03/05/2012 (Esta página foi modificada pela última vez à(s) 04h16min de 31 de março de 2012).

[27] Tom Pender. *UML a Bíblia*. Ed. Campus. 2004
[28] http://fabriciosanchez.com.br/2/wsdl-o-que-e-pra-que-serve-onde-utilizo/ acessado em 30/08/2012