Universidade Federal Fluminense
Instituto de Computação
Curso de graduação em Ciência da Computação

Ana Paula Lima Lucas
Helena de Fátima Maimputo Victor

DESENVOLVIMENTO DE UM SERVIÇO UTILIZANDO WEB SERVICE: UM ESTUDO DE CASO.

Niterói
2012

Ana Paula Lima Lucas
Helena de Fátima Maimputo Victor

DESENVOLVIMENTO DE UM SERVIÇO UTILIZANDO WEB SERVICE: UM ESTUDO DE CASO.

Dissertação apresentada ao Curso de Graduação em Ciência da Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Bacharel. Área de Concentração Engenharia de Software.

Orientadora: TERESA CRISTINA DE AGUIAR

Niterói
2012

Universidade Federal Fluminense
Instituto de Computação
Ciência da Computação

Ana Paula Lima Lucas
Helena de Fátima Maimputo Victor

DESENVOLVIMENTO DE UM SERVIÇO UTILIZANDO WEB SERVICE: UM ESTUDO DE CASO.

Niterói
2012

Ana Paula Lima Lucas
Helena de Fátima Maimputo Victor

DESENVOLVIMENTO DE UM SERVIÇO UTILIZANDO WEB SERVICE:
UM ESTUDO DE CASO.

Dissertação apresentada ao Curso de Graduação em Ciência da Computação da Universidade Federal Fluminense, como requisito parcial para obtenção do Grau de Bacharel. Área de Concentração Engenharia de Software

Aprovado em 16 de outubro de 2012.

BANCA EXAMINADORA

_____
Profa. D. Sc. TERESA CRISTINA DE AGUIAR – Orientadora
UFF

_____
Profa. D.Sc. ROSÂNGELA LOPES LIMA
UFF

_____
Prof. D.Sc. LEONARDO CRUZ DA COSTA
UFF

Niterói
2012

AGRADECIMENTOS

Em primeiro lugar agradecemos a DEUS, pois sem ELE eu não teria forças para vencer esta longa jornada. Aos meus pais Manoelino Lucas e Cosmea Lima Lucas por proporcionar educação, amor, dedicação e incentivo. Foram anos de luta, cujos esforços jamais serão medidos.

Agradeço também a todas as pessoas que, direta ou indiretamente, me ajudaram a vencer todas as batalhas. Agradeço em especial a minha amiga Helena de Fátima M. Victor por tudo que passamos juntas, alegrias e tristezas compartilhadas. (Ana Paula Lima Lucas)

Agradeço primeiramente a Deus, por ter me iluminado e ter me concedido saúde e inteligência, possibilitando a conclusão deste importantíssimo trabalho para a minha vida pessoal e profissional.

Em seguida, aos meus pais Agostinho Dianda e Morena Maimputo, que sempre me deram força e estímulo nos momentos mais difíceis.

Aos meus tios Sivivi Maimputo e Letícia Maimputo pelos conselhos e amizade nesta jornada.

Aos professores da Computação da Universidade Federal Fluminense pelos ensinamentos ao longo do curso. A minha preciosa colega e amiga Ana Paula Lima Lucas por trabalhar neste projeto comigo. Todo esforço, perseverança e disciplina valeram a pena. Aos amigos que a vida vem carregando, pela compreensão que mesmo na ausência, mais minha do que suas vocês não serão esquecidos.

Por fim, agradeço a professora Teresa, por ter me dado a oportunidade de realizar este trabalho. (Helena de Fátima Maimputo Victor)

EPÍGRAFE

“Arquive as autoridades que passaram em sua vida e dominaram seus conhecimentos. Desperte e crie o seu mundo! Com vivacidade e veemência busque métodos para instigar sua curiosidade. Você verá o seu mundo e sua vida com outros olhos.”

Mauro Nunes

RESUMO

Atualmente existe a necessidade de troca de informações entre sistemas e o presente trabalho aborda esse tema. Neste trabalho é realizado um estudo sobre a Arquitetura Orientada a Serviços e Web Services. Através do estudo do caso foi desenvolvido um Web Service em Java, para distribuição de notícias, seguindo um processo de desenvolvimento de serviços para reúso. Este processo é composto por três etapas: na primeira etapa foi feita a identificação do serviço e levantamento dos requisitos funcionais e não funcionais. Na segunda etapa especificamos as operações, as exceções e a descrição do serviço. E por último implementamos e testamos o serviço. Web Service torna-se uma escolha inteligente por sua interoperabilidade, permitindo a comunicação com aplicações desenvolvidas em diferentes plataformas de *hardware* e linguagens de programação através de uma rede local ou Internet. As aplicações se comunicam com o Web Service através de protocolos baseados em XML.

Palavras-chaves: Web Services e XML.

ABSTRACT

Currently there is a need for information exchange between systems and this paper addresses this topic. This work is a study on the Service Oriented Architecture and Web Services. Through the case study was developed a web service in Java, for news distribution, following a process of developing services for reuse. This process consists of three steps: the first step identifies the service and collects functional and non-functional requirements. In the second stage we specify the operations, exceptions and service description. And finally we implemented and tested the service. Web Service becomes a smart choice for its interoperability, enabling communication with applications developed on different hardware platforms and programming languages via a local network or Internet. The applications communicate with the Web Service via XML based protocols.

Keywords: Web Service, XML.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS

API	*Application Programming Interface*
CASE	*Computer Assisted Software Engineering*
HTTP	*Hypertext Transfer protocol*
REST	*Representational State Transfer*
SOA	*Service Oriented Architecture*
SOAP	*Simple Object Access Protocol*
SQL	*Structure Query Language*
UDDI	*Universal Description Discovery and Integration*
URL	*Uniform Resource Locator*
W3C	*World Wide Web Consortium*
WDSL	*Web Services Description Language*
XML	*eXtensible Markup Language*

SUMÁRIO

CAPÍTULO 1 – INTRODUÇÃO

1.1 – MOTIVAÇÃO

A motivação principal para elaborar este trabalho foi obter conhecimento detalhado do desenvolvimento de um Web Service e entender seu mecanismo de funcionamento. Atuando no mercado de trabalho, sentimos a necessidade de aprofundar nosso conhecimento sobre o tema, pois além de ser uma tecnologia atual, é um tema que particularmente desperta grande interesse nosso.

Atualmente devido às novas exigências de negócios, diversas organizações têm optado pela arquitetura orientada a serviços e utilizado os Web Services como uma tecnologia de implementação desta arquitetura.

Web Services oferecem uma solução para questões como a comunicação, padronização e a interoperabilidade.

Interoperabilidade é a capacidade de um aplicativo se comunicar com qualquer outro aplicativo, independente do *hardware* no qual é distribuído, no ambiente operacional no qual o aplicativo é executado ou da linguagem de programação utilizada para desenvolvê-lo. (Kumar, 2012)

Um Web Service provê dados e serviços para outras aplicações ou serviços, ou seja, web services conectam aplicações diretamente com outras aplicações. (ABINADER, 2006)

O W3C (World Wide Web Consortium) define "Web Service", como "um sistema de software projetado para suportar interação máquina-máquina através de uma rede". Ele tem uma interface descrita em formato processável por máquina chamada WSDL (Web Services Description Language). Outros sistemas interagem com Web Services na forma prescrita pela sua descrição utilizando mensagens SOAP (Simple Object Access Protocol), geralmente transmitidas utilizando protocolo HTTP.

1.2 – OBJETIVO

Este trabalho possui dois objetivos. O primeiro é resumir os conhecimentos adquiridos durante as etapas de estudo sobre a arquitetura orientada a serviços e Web Services. E o segundo objetivo é a elaboração de um Web Service em Java, destinado à distribuição de notícias. Para sua elaboração é seguido um processo de desenvolvimentos de serviços para reúso.

1.3 - ORGANIZAÇÃO DO TRABALHO

Esta monografia é composta de seis capítulos, cada um deles abordando assuntos específicos sobre o tema proposto. Os capítulos estão organizados da seguinte forma:

Capítulo 1: Apresenta a motivação para realização do trabalho e trata também dos objetivos e de como foi organizada esta monografia.

Capítulo 2: Apresenta a Arquitetura Orientada a Serviços, destacando as entidades fundamentais e características de serviço.

Capítulo 3: Nesse capitulo serão apresentados conceitos que envolvem Web Services, apresentando definições, suas características e a relação entre os seus principais elementos.

Capitulo 4: Descreve o processo de desenvolvimento de serviços apresentado em Sommerville (2012).

Capítulo 5: É apresentado o estudo de caso, na qual, será implementado um Web Service.

Capítulo 6: E por fim, este capítulo apresenta a conclusão.

CAPÍTULO 2 – ARQUITETURA ORIENTADA A SERVIÇOS

Este capítulo apresenta a Arquitetura Orientada a Serviços, destacando suas entidades fundamentais e características de serviço.

Um sistema distribuído segundo a definição apresentada em Coulouris (2010), diz: "coleção de computadores autônomos interligados através de uma rede de computadores e equipados com software que permita o compartilhamento dos recursos do sistema: hardware, software e dados".

Há vários tipos de arquiteturas de sistemas distribuídos, como por exemplo, a Arquitetura cliente-servidor e a arquitetura orientada a serviços (SOA - *Service Oriented Architecture*).

Uma arquitetura orientada a serviços tem como componente fundamental o conceito de serviços. O conceito de serviço é definido para se referenciar a um componente de software binário baseado em um contrato, que seria uma definição ou descrição de ações a serem executadas. (COUTINHO, 2004)

2.1 - SERVIÇOS

Um serviço pode ser definido como: Um componente de software reusável não firmemente acoplado que engloba a funcionalidade discreta que pode ser distribuída e acessada por meio de um programa. Segundo Turner (2003), a essência de um serviço é que o fornecimento dos serviços é independente da aplicação que usa o serviço.

Por exemplo, uma empresa pode ter uma aplicação de processamento de pedidos que utilize serviços como verificação de crédito do cliente e pesquisa de itens derivados de várias fontes dentro e fora da empresa. Estes mesmos serviços podem ser utilizados por outras aplicações.

Os serviços devem ser implementados apenas uma vez, de forma que possam ser reutilizáveis. E assim todos os sistemas que precisem de certa funcionalidade poderiam utilizar o mesmo serviço. (JOSUTTIS, 2007)

Os serviços também podem ser compostos, ou seja, um serviço pode chamar outro, permitindo assim que uma funcionalidade maior possa ser composta por outros serviços.

Um serviço deve ser independente de outro serviço. E deve possibilitar o acesso e uso da mesma aplicação ou de outras aplicações. Para isso, o serviço deve possuir uma interface padronizada que descreva os parâmetros técnicos, limitações e políticas que definem as características e as maneiras de acesso a ele, além de definir quais os possíveis protocolos de segurança necessário para acessá-lo.

Erl (2005) identifica três tipos fundamentais de serviços:

- Serviços de utilidades: Um exemplo é o serviço de conversão de moeda, que pode ser acessado para processar a conversão de uma para outra moeda.

- Serviços de negócios: Um exemplo de um serviço de negócio em uma universidade é o registro de estudantes em um curso.

- Serviços de coordenação: Um exemplo de um serviço de coordenação de serviços em uma empresa é um serviço de pedido que permite que os pedidos sejam feitos para fornecedores, mercadorias aceitas e pagamentos feitos.

2.2 – ENTIDADES FUNDAMENTAIS

Segundo Abinader (2006), o mecanismo de percurso de um serviço em uma Arquitetura Orientada a Serviços pode ser visto nas figuras 1 e 2. Na utilização dos serviços existem três entidades fundamentais:

- Provedor de serviço: Oferece um serviço pela definição de sua interface e pela implementação da funcionalidade do serviço. Para assegurar que o serviço possa ser acessado por usuários de serviços externos, o provedor de serviços cria uma entrada em um diretório de serviços, que inclui informações sobre o serviço e o que ele faz.

- Diretório de serviços: É onde são publicadas descrições dos serviços. Funciona como um catálogo onde são incluídos e podem ser procurados serviços específicos;

- Consumidor, cliente ou solicitante de serviços: Entidade que procura por serviço no diretório, e ao encontrar esse serviço, dirige-se ao provedor do serviço para consumi-lo, ou seja, vincular esse serviço a sua aplicação. Isso significa que a aplicação do solicitante inclui código para chamar o serviço e processar os resultados da chamada. Alguns exemplos de consumidores de serviços são aplicativos de usuário final, portais e sistemas internos.

Essas entidades devem executar três operações básicas entre si:

- Operação de publicação, de registro de serviço, dos métodos e da interface no diretório.

- Operação de busca, de consulta do diretório pelo consumidor que está interessado em encontrar o serviço.

- Operação de consumo do serviço a partir do consumidor que pode usufruir o serviço disponibilizado pelo provedor.

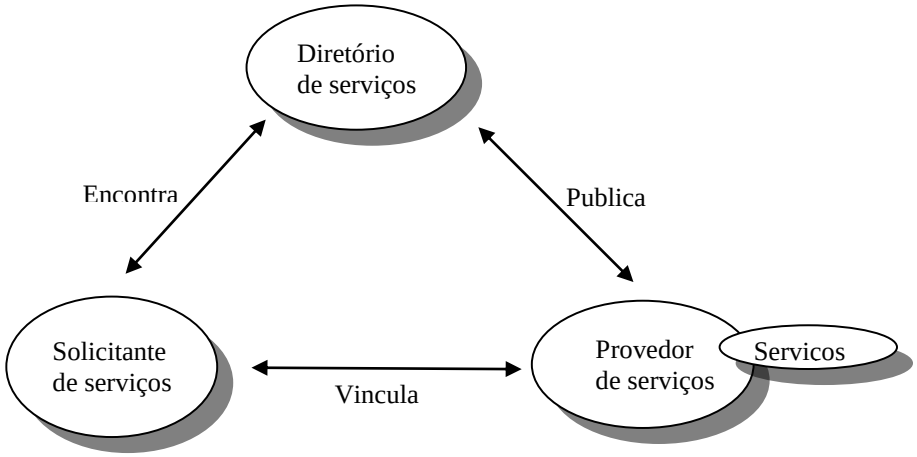


Figura 1: Arquitetura Orientada a Serviços
Elaboração própria baseada em Sommerville (2011)

Segundo Sommerville (2011), algumas características de serviços são:

- Os serviços podem ser oferecidos por qualquer provedor de serviços dentro ou fora de uma organização.

- As empresas podem criar aplicações por meio da integração de serviços de vários provedores.

- O provedor de serviços torna públicas as informações sobre o serviço de maneira que os usuários autorizados possam usá-lo.

- As aplicações podem retardar a vinculação dos serviços até que eles sejam implantados ou executados.

- A criação conveniente de novos serviços é possível. Um provedor de serviços pode reconhecer novos serviços que podem ser criados pela vinculação de serviços existentes de maneiras inovadoras.

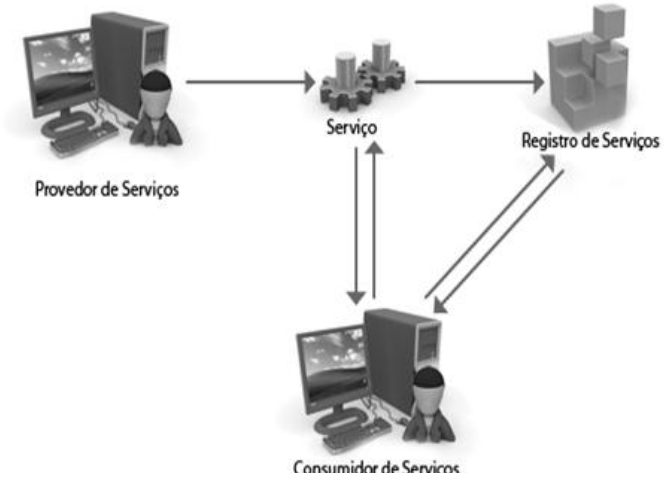


Figura 2: Papéis e funções
Fonte: Abinader (2006)

CAPÍTULO 3 - WEB SERVICES

Nesse capitulo serão apresentados conceitos que envolvem Web Services, apresentando definições, suas características e a relação entre os seus principais elementos.

McGoven (2003) diz que é importante para desenvolvedores de serviços compreender os conceitos de SOA (*Service Oriented Architecture),* de modo que eles possam fazer o uso mais eficaz de Web Services em seu ambiente, pois a Arquitetura Orientada a Serviços pode ser implementada através de Web Service. A figura 3 exemplifica esta ideia.

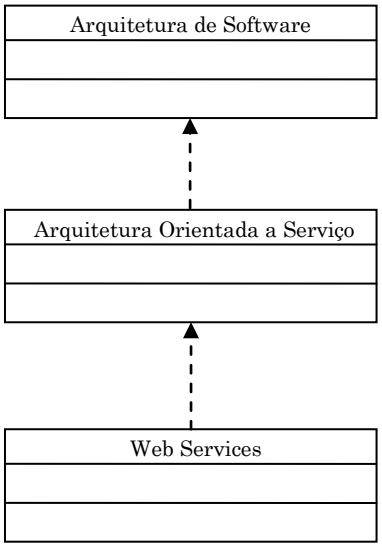


Figura 3: Relação entre SOA e Web Services

3.1 – CONCEITOS BÁSICOS

Web Services podem ser definidos como programas modulares, independentes e auto-descritivos, que podem ser localizados e invocados através da internet. (W3C, 2009)

Um Web Service é baseado em XML (*eXtensible Markup Language*), o que facilita a integração e colaboração interna e com parceiros.

Segundo W3C Working Group (2004a), os três padrões fundamentais que possibilitam comunicações entre aplicações através de Web Services são:

1) SOAP (*Simple Object Access Protocol*): Provê uma forma de representar e estruturar os dados que representam as mensagens trocadas entre as aplicações em um ambiente distribuído.

2) WSDL (*Web Services Description Language*): Descreve a interface pública de um serviço, disponibilizando informações para que seus métodos possam ser acessados.

3) UDDI (*Universal Description Discovery and Integration*): Possibilita que serviços em toda a Internet sejam registrados e possam ser encontrados por outros serviços.

Todos estes padrões, descritos com mais detalhes na seção 3.3, são baseados em XML – uma linguagem de marcação legível por humanos e máquinas, responsável pela descrição das informações.

Atualmente há dois grupos de Web Services: baseados em SOAP e REST (*Representational State Transfer*).

Neste trabalho é abordado Web Service baseado em SOAP.

Web Services empregam SOAP para troca de mensagens, contêm metadados que descrevem o serviço disponibilizado e possuem diretório integrado para descrição e localização de Web Services. (ABINADER, 2006)

Segundo Kalim (2009), características descritas a seguir, distinguem Web Services de outros sistemas de software distribuídos. :

- Infraestrutura aberta: Web Services que usam protocolos-padrão da indústria e independentes do fornecedor como HTTP e XML são onipresentes e bem compreendidos. Web Services suportam 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: O serviço e seus clientes não precisam ser escritos na mesma linguagem. Transparência de linguagem é a chave para a interoperabilidade de Web Services; isto é, a habilidade de Web Services e solicitantes interagirem apesar das diferenças nas linguagens de programação, bibliotecas de suporte e plataformas.

- *Design* modular: Webs Services são designados para serem modulares em design, para que novos serviços possam ser gerados através da integração e agrupamento de serviços existentes. (KALIM, 2009)

A figura 4 apresenta características importantes dos Web Services.

Figura 4: Visão Geral dos Web Services

Elaboração própria baseada em Abinader (2006)

As arquiteturas de aplicação de Web services são arquiteturas não firmemente acopladas nas quais as ligações de serviços podem mudar durante a execução. Assim, versões equivalentes, mas diferentes de um serviço, podem ser executadas em momentos diversos. Para ilustrar como as aplicações podem ser organizadas, Sommerville (2011) apresenta o seguinte exemplo:

Um sistema de informações de um automóvel fornece aos motoristas informações sobre o clima, condições de tráfego e informações locais. O sistema é ligado ao aparelho de rádio do automóvel que apresenta as informações com um sinal de uma emissora de rádio específica. O automóvel é equipado com receptor GPS (sistema de posicionamento global) para informar sua posição e, baseado nessa posição, o sistema acessa serviços de informações. As informações podem ser fornecidas no idioma especificado pelo motorista.

A figura 5 ilustra uma possível organização para tal sistema. O software do automóvel inclui cinco módulos. Estes cuidam da comunicação com o motorista, com o receptor GPS informando a posição do automóvel e com o aparelho de rádio do automóvel. Os módulos Transmissor e Receptor cuidam de todas as comunicações com os serviços externos.

O automóvel se comunica com um serviço de informações móvel fornecido externamente que agrega informações de outros serviços que fornecem informações sobre o clima, informações sobre tráfego e recursos locais. Provedores diferentes em localidades diferentes fornecem esses serviços e o sistema do automóvel usa um serviço de descobrimento para localizar o serviço de informações apropriado e se conectar a ele. O serviço de

descobrimento é também usado pelo serviço de informações móvel para vincular os serviços apropriados de clima, tráfego e recursos. Os serviços trocam mensagens SOAP que incluem as informações de GPS usadas pelos serviços para selecionar as informações adequadas. As informações agregadas retornam ao automóvel por meio de um serviço que traduz o idioma das informações no idioma do motorista.

Quando o automóvel se move, o software interno usa o serviço de descobrimento para encontrar o serviço de informações mais apropriado e se conecta a ele.

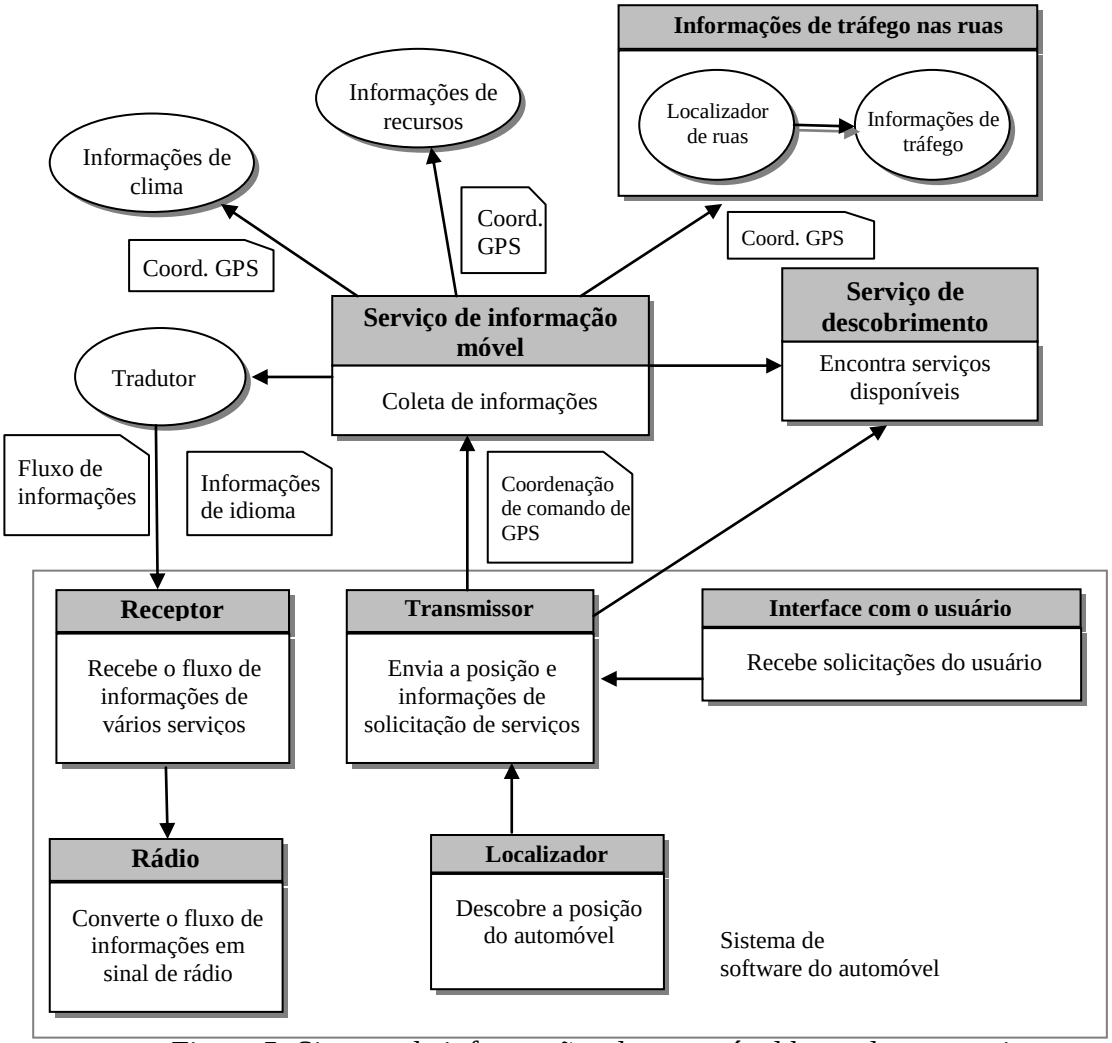


Figura 5: Sistema de informações de automóvel baseado em serviços
Elaboração própria baseada em Sommerville (2011)

3.2 - MODELO DE IMPLEMENTAÇÃO DE WEB SERVICES

A seguir são descritos os principais passos do modelo de implementação de Web Services. Na figura 6, apresentamos de modo simplificado como e onde cada um dos passos

descritos ocorre. O número apresentado após a descrição dos passos tem a função de facilitar a associação do texto com a imagem.

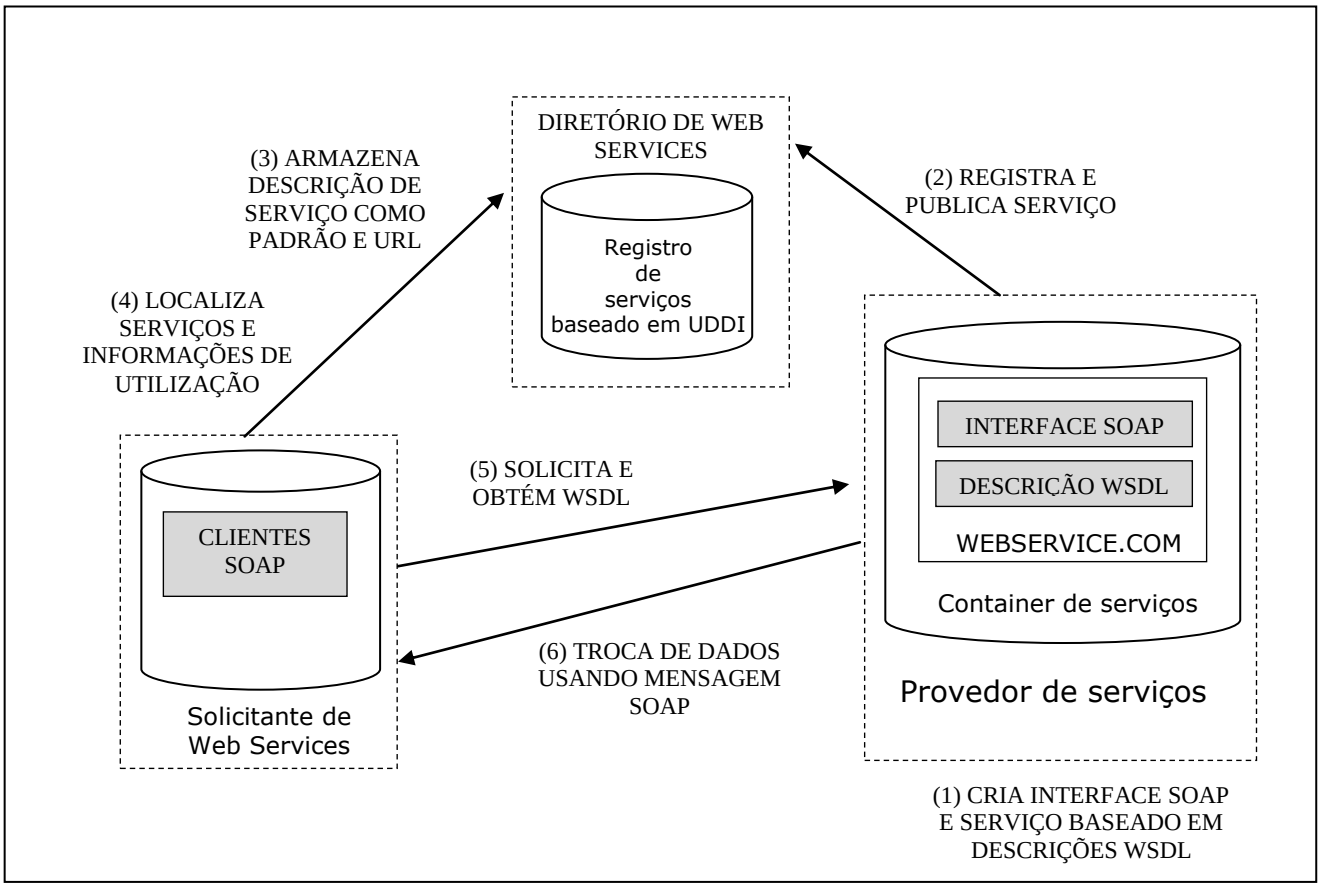


Figura 6: Implementação do Web Service
Elaboração própria baseada em Abinader (2006)

1) O provedor de serviço cria os Web Services tipicamente como serviços baseados na interface SOAP e disponibiliza os Web Services em um *container* de serviço, e os torna acessíveis através da Internet. O provedor de serviço também disponibiliza a descrição dos Web Services no padrão WSDL para acesso ao serviço, que define como o cliente e container devem interagir de modo consistente, expressando claramente a identificação e localização do serviço, operações disponíveis e o modelo de comunicação implementado [1];

2) Após a implementação dos Web Services, o provedor de serviço registra os mesmos, com base na descrição de serviço WSDL, no diretório de serviço, tipicamente um registro UDDI [2];

3) Diretório de registro de Web Services, um registro UDDI armazena a descrição dos serviços em formato padronizado, disponibilizando URL para a localização do WSDL no ambiente do provedor dos Web Services [3];

4) Solicitante do serviço pode então localizar os Web Services desejados a partir de pesquisas no registro UDDI. O solicitante dos Web Services obtém a informação necessária e o URL para identificar e acessar o provedor de Web Services [4];

5) Empregando as informações obtidas no registro UDDI, o solicitante do serviço invoca o provedor dos Web Services e solicita a descrição dos serviços no padrão WSDL para os serviços registrados. Após isto, o solicitante dos serviços cria um Proxy cliente para que a aplicação tenha acesso aos Web Services estabelecendo comunicação como o provedor através de SOAP [5];

6) O solicitante dos Web Services comunica-se com o provedor e inicia a troca de dados ou mensagens, invocando os serviços disponíveis no container [6];

3.3 – PADRÕES DE TECNOLOGIAS DE WEB SERVICES

3.3.1 – XML (EXTENSIBLE MARKUP LANGUAGE)

Um esforço coordenado pelo W3C (World Wide Web Consortium), comunidade internacional que desenvolve padrões com o objetivo de garantir o crescimento da web, foi movido visando oferecer uma nova linguagem que pudesse satisfazer às necessidades de interoperabilidade, escalabilidade e flexibilidade, permitindo-se fácil extensão.

A W3C assegura que os dados estruturados serão uniformes e independentes de sistemas tanto em hardware quanto em software. O XML é um formato padrão que pode exemplificar o conteúdo, as semânticas e os esquemas de uma enorme variação de sistemas desde os mais simples até os mais complexos.

Uma característica importante é que uma vez que o dado seja recebido pelo cliente, o mesmo pode ser manipulado, alterado e visualizado sem a necessidade de acionar novamente o servidor que o enviou. Dessa forma, os servidores têm menor sobrecarga de requisições, o que reduz o processamento da máquina, reduzindo também o tráfego de dados pela rede para as comunicações entre cliente e servidor.

Um documento em XML compreende a descrição de dados, tornando-se possível seu processamento por uma aplicação. O XML tem sido cada vez mais utilizado por desenvolvedores de aplicações devido ao suporte que ele oferece tanto a interoperabilidade quanto funcionalidade da Web. Trata-se de uma linguagem baseada em texto a qual permite a qualquer pessoa escrever um código em XML, sendo ele, facilmente, compreensível às pessoas quanto manipulável pelos computadores. (MOULTIS, 2000)

Em um documento XML os dados constam na forma de texto limitados por *tags* que descrevem a informação.

XML é uma meta-linguagem de marcação de dados (*meta-markup language*). XML não têm um conjunto fixo de *tags* e elementos para todas as áreas. Ela permite que os desenvolvedores definam os elementos necessários a um área.

Segundo Daum (2002), tal linguagem é considerada como ferramenta de padronização, já que esta utiliza marcações, isto é, *tags* que podem ser definidas pelo usuário. A partir destas *tags* definidas, o usuário pode delimitar seu texto da forma que bem entender, tornando-o assim mais legível e mais interpretável. Isso ocorre porque, geralmente, o usuário utiliza palavras-chave do texto que pretende se estruturar em XML. Para delimitá-lo, estas palavras-chave dão nome às novas *tags* que estão sendo criadas.

Outro fator que determinou a linguagem XML como uma linguagem de padronização é o fato de ela possuir compatibilidade com mais de um conjunto de caracteres. Este fato se torna uma vantagem em relação às outras linguagens que trabalham apenas com padrões americanos, assim como o ASCII (*American Standard Code* -Código Americano Padrão).

Outra utilização encontrada por usuários de todo o mundo, para XML, é o armazenamento de informações. Muitos usam esta linguagem para armazenamento, já que esta ferramenta possui características para estruturação de dados utilizando uma arquitetura que segue o modelo de dados hierárquico. Tais características permitem que dados sejam organizados e trabalhados de forma simples. (ANDERSON, 2001)

Torna-se imperativo ainda acrescentar que XML é uma linguagem simples, possui um conjunto de estruturas de dados ricas, permite a troca e exibição de conteúdo de bases de dados e pode ser utilizada como formato para troca de mensagens na comunicação entre aplicações.

A grande maioria dos documentos XML centrados em dados possuem tempo de vida muito curto, pois são produzidos a partir de dados estruturados que podem ser modificados a cada milissegundo, de modo que a aplicação pode produzir o resultado de uma consulta sobre uma tabela em um banco de dados e entregá-lo como documento XML. Com esta flexibilidade, XML tornou-se o padrão para o transporte de dados estruturados, conteúdo e formato para dados que representam documentos em meio eletrônico.

A figura 7 demonstra graficamente como XML posiciona-se na base dos três principais padrões adotados para constituir a tecnologia de Web Services.

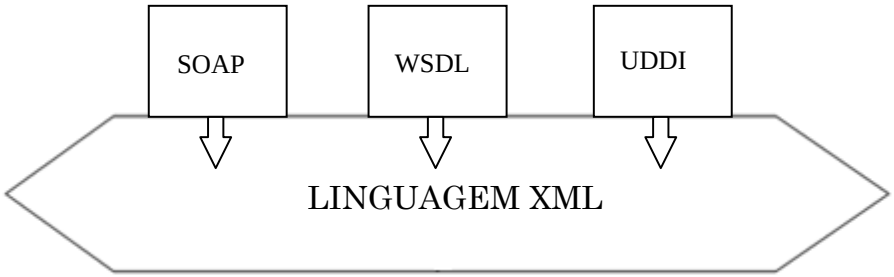


Figura 7: Linguagem XML como base de Web Services
Fonte: Abinader (2006)

Segundo Abinader (2006), os documentos XML podem ser disponibilizados e utilizados nas mais diversas aplicações, demonstrando o porquê de sua importância na interoperabilidade de aplicações. O XML possibilita que sejam criadas soluções tanto do lado cliente quanto do servidor. Por exemplo, no lado cliente, os documentos XML podem ser utilizados para a apresentação de conteúdo personalizado.

Do lado dos servidores, seu impacto é mais claro, e isso é mais nítido, já que os documentos que obedecem a esse padrão consistem no coração dos *Web Services* que, no caso, são as mensagens. Esse sistema de mensagens é a possibilidade de intercâmbio de dados entre aplicações e computadores, no qual é estabelecido um formato e regras claras para os pares.

3.3.2 – SOAP (*Simple Object Access Protocol*)

Segundo W3C (2000), o SOAP é um protocolo leve para troca de mensagens em um ambiente distribuído e descentralizado. SOAP é um protocolo independente de plataforma e independente de implementação. Permite baixo acoplamento entre requisitante e provedor e

permite comunicação entre serviços de diferentes organizações. É um protocolo baseado em XML que consiste em três aspectos:

1) O modo como a mensagem XML é estruturada;
2) Convenções que representam a chamada remota de procedimento (RPC – *Remote Procedure Call*). RPC é uma tecnologia de comunicação entre processos que permite a um programa de computador chamar um procedimento em outro espaço de endereçamento (geralmente em outro computador, conectado por uma rede);
3) A comunicação com o protocolo HTTP, para garantir que a mensagem XML seja transportada corretamente e convenções para representar a ocorrência de erros que devem ser informados de volta ao emissor da mensagem SOAP. (ABINADER, 2006)

A unidade básica de transmissão SOAP é o elemento XML envelope, responsável por descrever o conteúdo da mensagem fornecendo informações sobre como processar esse conteúdo. O envelope possui dois elementos filhos: header, opcional, que possui informações de controle e o *body* que contém o corpo da mensagem XML conforme mostrado na figura 8.

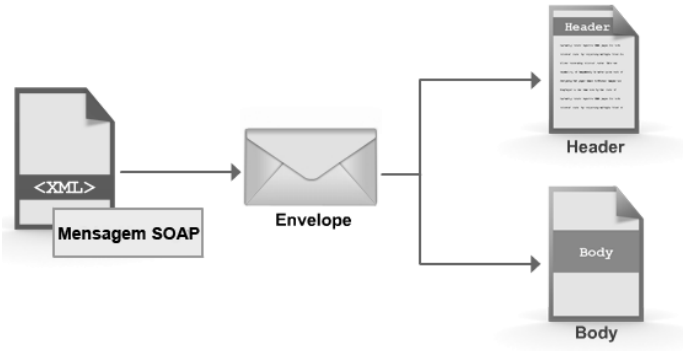


Figura 8: Representação XML do SOAP
Fonte: Abinader (2006)

Vimos que uma mensagem SOAP é codificada usando um documento XML. Na estrutura deste documento encontramos os seguintes elementos:

Envelope: É o elemento raiz, ou melhor, central e obrigatório. Este delimita a mensagem SOAP, isto é, indica ao requisitante onde começa e termina a mensagem. Com este elemento, o receptor ou emissor sabe que está lidando com uma mensagem SOAP, sendo este seu principal objetivo. Funciona como um recipiente para os elementos Body e Header, mencionados a seguir.

Header: Se presente, o elemento deve estar localizado, obrigatoriamente, após o campo Envelope. Nele encontramos as informações para processamento da mensagem, por exemplo, autenticação, autorização e outros, além do atributo ator, que contém o próximo destino intermediário no caminho da mensagem. (KUMAR, 2012)

Body: De acordo com W3C (2000), este elemento é obrigatório, e deve vir imediatamente após o elemento Header, se presente. Caso contrário, ele deve ser o primeiro elemento depois do elemento Envelope. Segundo Abinader (2006), a mensagem transmitida do provedor para o requisitante ou vice-versa, via protocolo SOAP, vai contida neste elemento. Existem dois estilos possíveis de mensagens: estilo RPC, e o estilo documento XML completo. No estilo RPC, são descritos detalhes sobre a procedure (método ou operação) a ser chamada e os parâmetros adequados, empregando-se o modo de codificação negociado entre as partes que se comunicam. O estilo documento XML completo, como o próprio nome informa, deve conter um documento XML que está sendo trocado entre as partes. SOAP define um atributo opcional para o campo Body chamado Fault. Este atributo tem como função relatar erros resultantes na execução de mensagens ao emissor. A figura 9 ilustra a estrutura de uma mensagem SOAP.

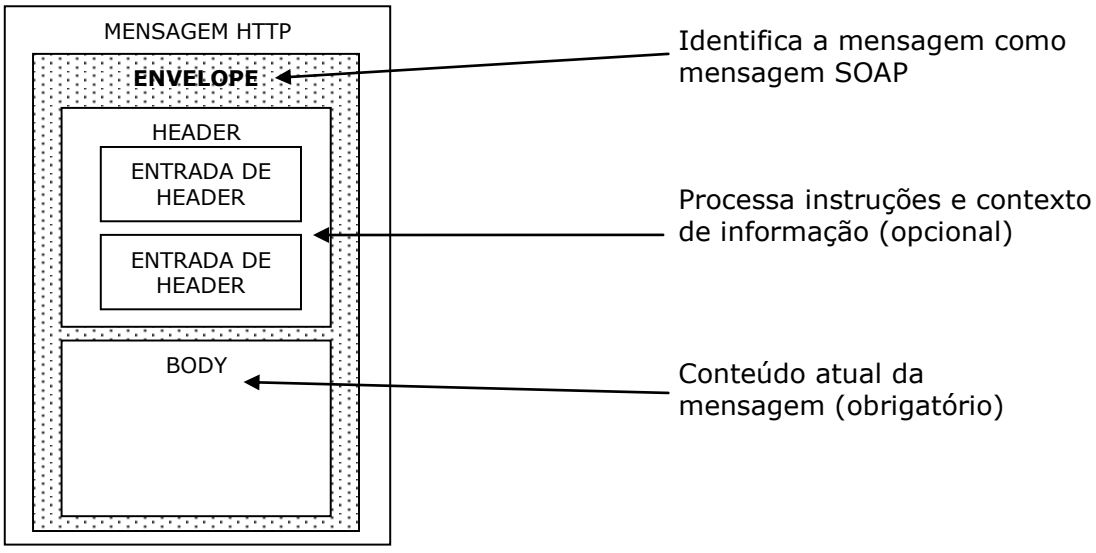


Figura 9: Estrutura da mensagem SOAP
Fonte: Abinader (2006)

3.3.3 - WSDL (*Web Services Description Language*)

O W3C (World Wide Web Consortium) oferece a seguinte definição para WSDL: “WSDL constitui-se em formato XML para descrição de serviços de rede, como conjunto de

pontos de acesso, disponibilizando mensagens com informações que podem ter conteúdo orientado a documento ou conteúdo orientado a *procedure*. Operações para obter mensagens, e as mensagens em si, são descritas de forma abstrata e ligadas concretamente ao protocolo de rede e ao formato da mensagem para definir o ponto de acesso, relacionando os pontos de acesso concreto com pontos de acesso abstratos (serviços).”

É imprescindível que a tecnologia WSDL permita que o cliente interessado em utilizar o Web Service possa fazê-lo de forma automática, sem que seja necessário intervenção ou contato com o autor do mesmo. (ABINADER, 2006)

Segundo Sommerville (2011), a especificação WSDL define três pontos sobre um Web Service. Ela define o que o serviço faz, como ele se comunica e onde encontrá-lo:

A parte “o que” de um documento WSDL, chamada interface, especifica quais operações o serviço apóia e define o formato das mensagens enviadas e recebidas pelo serviço.

A parte “como” do documento WSDL, chamada ligação (*binding*), mapeia uma interface abstrata para um conjunto concreto de protocolos. A ligação especifica os detalhes técnicos de como se comunicar com um Web service.

A parte “onde” do documento WSDL, chamada serviço, descreve onde localizar um Web service específico implementado.

Minimamente, o consumidor de Web services precisará conhecer a assinatura do serviço (o que entra e o que sai), sua localização e o protocolo a ser usado para enviar a invocação. Um documento WSDL fornece estas informações organizadas em duas seções lógicas: descrição abstrata e descrição concreta. (ABINADER, 2006)

A parte de descrição abstrata é composta por quatro elementos XML:

- *Types*: especifica e lista todos os tipos definidos pelo usuário invocados pelo Web service. A seção *types* pode estar vazia, quando o serviço usa apenas os tipos de dados simples como string e long. Mas essa seção pode conter um diretriz *import* para o documento XSD (XML *Schema Definition*) associado, que define tipos de dados complexos.
- *Messages*: contém parâmetro de entrada e saída para o serviço e descreve as diferentes mensagens trocadas pelo serviço. As mensagens são construídas a partir de tipos de dados definidos na seção *types*. Cada elemento *messages*

possui um atributo *name* que deve conter o nome da mensagem, de modo que o mesmo seja único em todo documento WSDL. Cada elemento *messages* pode conter zero ou mais elementos *part*, que representam os atributos da mensagem, os parâmetros da *procedure*. Cada elemento *part* possui o atributo *name*, que deve ser único e o atributo *type*, que deve associar o elemento *part* a algum tipo definido na seção *types*.

- *Operation*: representa a interação particular com o serviço e descreve as mensagens de entrada e saída e exceções possíveis e permitidas durante a interação.
- *PortTypes*: descreve o conjunto de operações que pode ser executado pelo serviço.

Já a parte da descrição concreta é composta por dois elementos XML:

- *Binding*: especifica a ligação de cada operação descrita na seção do elemento *portTypes*, associando a descrição abstrata de *portTypes* (operações *portTypes*, mensagens e tipos de dados) com o protocolo de rede.
- Service: o documento WSDL deve descrever onde o serviço está disponível ou publicado. A associação entre o elemento *binding* e o endereço de rede onde o serviço pode ser encontrado é definido pelo subelemento de service, o elemento port.

A separação lógica da informação abstrata (como métodos, parâmetros e mensagens de erro) da informação que pode mudar com o tipo de implementação (protocolo de transporte, endereço de rede) permite o reuso das definições abstrata dos serviços através de diferentes implementações do mesmo. Como exemplo podemos citar um mesmo serviço exposto como baseado em HTTP e em SMTP (*Simple Mail Transfer Protocol*). Em ambos os casos, o serviço desempenha as mesmas funções, assim a descrição abstrata e os métodos do serviços permanecem os mesmos, podendo ser reutilizados. A figura 11 demonstra, de modo simplificado, a representação conceitual do documento WSDL. (ABINADER, 2006)

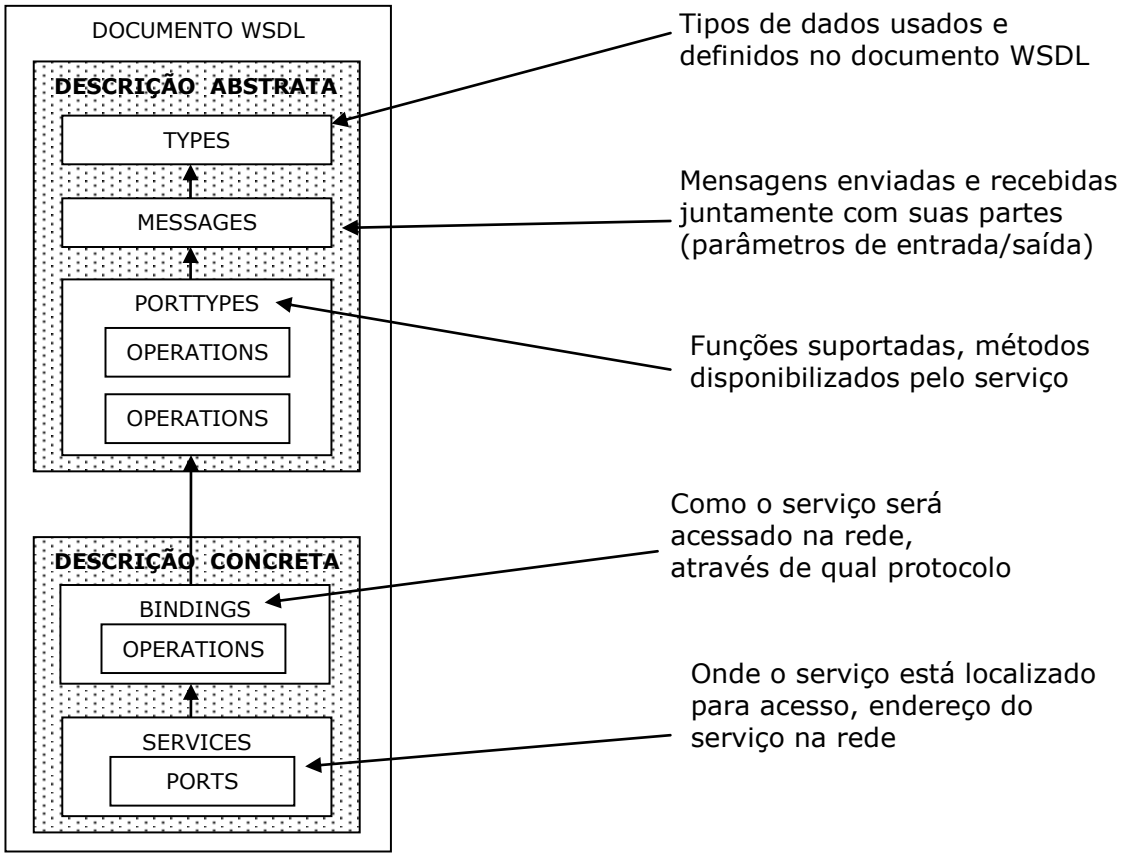


Figura 11: Representação conceitual do documento WSDL
Fonte: Abinader (2006)

3.3.4 – UDDI (*Universal Description Discovery and Integration*)

O padrão descrição, descoberta e integração universais define os componentes de uma especificação de serviços que pode ser usada para descobrir a existência de um serviço. Esses componentes incluem informações sobre o provedor de serviços, os serviços fornecidos e a localização da descrição de serviços (usualmente expressa em WSDL).

Conforme mostrado na Figura 12, o provedor de serviços responsável por manter o serviço cria a descrição do Web Service (WSDL) e publica a mesma no repositório UDDI. O Cliente faz uma busca no repositório, onde obtém o WSDL do serviço, a partir do qual cria um cliente e finalmente comunica-se com o serviço.

A descoberta de serviços usando um mecanismo de pesquisa-padrão para procurar descrições WSDL devidamente comentadas é, atualmente, a abordagem preferida para descobrir serviços externos. (SOMMERVILLE, 2011)

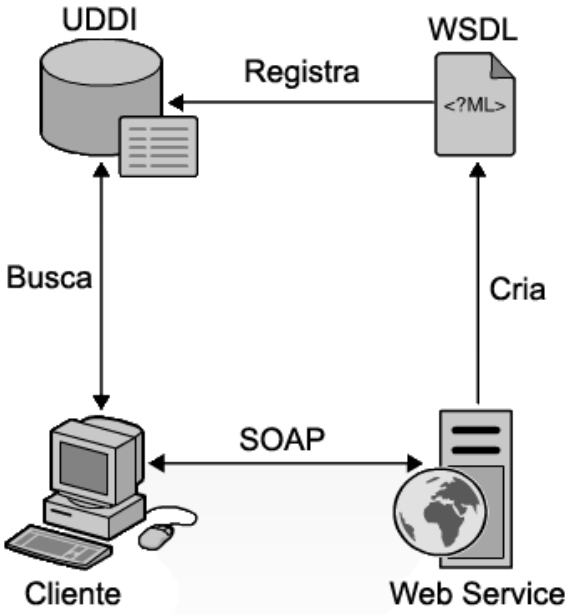


Figura 12: Visão geral do UDDI (ABINADER, 2006)

Então, resumidamente, poderíamos dizer que a figura 13, já apresentada anteriormente, ilustra como os Web Services são usados. Os provedores de serviços projetam e implementam serviços e os especificam em WSDL. Eles também publicam informações sobre esses serviços em um registro de acesso geral usando o padrão de publicação UDDI. Os solicitantes de serviços, que desejam fazer uso de um serviço, buscam o registro UDDI para descobrir a especificação desse serviço e para localizar um provedor de serviços. Eles podem então unir as suas aplicações para um serviço específico e se comunicar com ele, usando SOAP. (SOMMERVILLE, 2011)

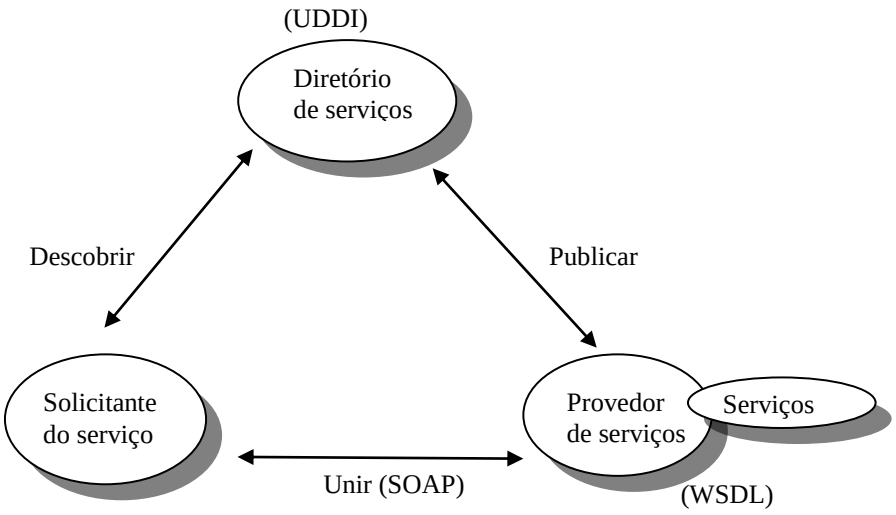


Figura 13: Arquitetura orientada a serviços
Elaboração própria baseada em Sommerville (2011)

CAPÍTULO 4 – UM PROCESSO PARA DESENVOLVIMENTO DE SERVIÇOS

Neste capítulo é apresentada a engenharia de serviços, um processo de desenvolvimento de serviços para reúso, direcionado para aplicações orientadas a serviços. Esse processo é apresentado em Sommerville (2011).

A engenharia de serviços tem que assegurar que o serviço representa uma abstração reusável que poderia ser útil em diferentes sistemas. Deve projetar e desenvolver funcionalidades úteis associadas com essas abstrações e deve também assegurar que o serviço seja robusto e confiável de modo a operar confiavelmente em diferentes aplicações.

O desenvolvimento de software usando serviços baseia-se na ideia de compor e configurar serviços para criar novos serviços compostos. Estes podem ser interligados com uma interface de usuário implementada em um browser para criar uma aplicação Web ou podem ser usados como componentes em alguma outra composição de serviço. Os serviços envolvidos na composição podem ser especialmente desenvolvidos para a aplicação. Atualmente, muitas empresas estão convertendo suas aplicações corporativas em sistemas orientados a serviços, em que um bloco básico de construção de aplicações é um serviço, e não um componente. Isso abre a possibilidade de reúso mais generalizado.

Segundo Sommerville (2011), há três estágios lógicos no processo de engenharia de serviços (figura 14). São eles:

- Identificação de serviço candidato – Identifica-se possíveis serviços que poderiam ser implementados e define-se os requisitos de cada serviço.
- Projeto de serviço – Projeta-se a lógica, mensagens e as interfaces de serviços WSDL.
- Implementação e implantação de serviço – Cada serviço é implementado e testado, sendo disponibilizado para uso.

4.1 – IDENTIFICAÇÃO DE SERVIÇO CANDIDATO

Como toda organização tem grande quantidade de processos há, portanto, muitos serviços possíveis que podem ser implementados. A identificação do serviço candidato envolve a compreensão e a análise desses processos para decidir quais serviços reusáveis são necessários para apoiar os processos.

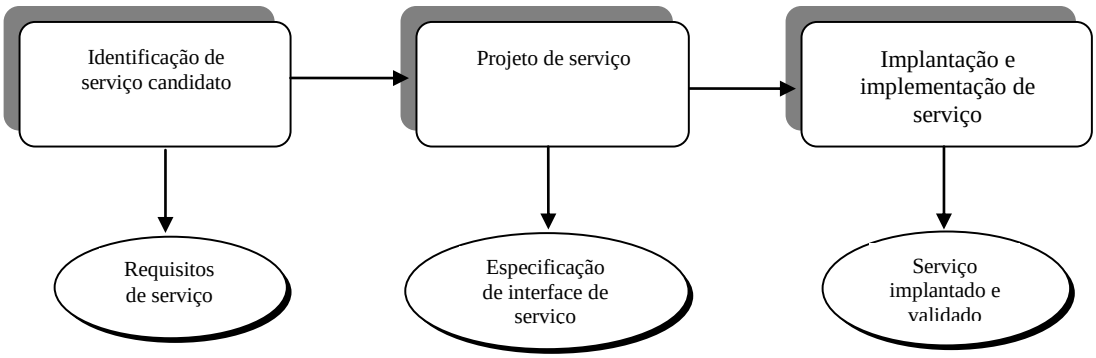


Figura 14: Processo de engenharia de serviços
Elaboração própria baseada em Sommerville (2011)

O objetivo na identificação de serviço candidato deve ser identificar serviços logicamente coerentes, independentes e reusáveis. A classificação de Erl (2005) é útil a esse respeito quando sugere como descobrir serviços reusáveis olhando as entidades e os processos.

A saída do processo de seleção de serviços candidatos é um conjunto de serviços identificados e requisitos associados a esses serviços. Os requisitos funcionais de serviços devem definir o que o serviço deve fazer. Os requisitos não funcionais devem definir, por exemplo, os requisitos de proteção, desempenho e disponibilidade do serviço.

4.2 - PROJETO DE SERVIÇO

Uma vez selecionados os serviços candidatos, o próximo estágio no processo da engenharia de serviço é projetar as interfaces de serviço. Isso envolve a definição de operações associadas com o serviço e seus parâmetros.

É importante também planejar como as operações e as mensagens de serviço podem ser projetadas para minimizar o número de troca de mensagens que deve ocorrer para completar a solicitação de serviço. É importante assegurar que o maior número possível de informações é passado para o serviço em uma mensagem antes de requerer interações síncronas de serviço.

Segundo Sommerville (2011), existem três estágios para projeto de interface de serviço:

- Projeto de interface lógica no qual se identifica as operações associadas com o serviço, as entradas e saídas dessas operações e as exceções associadas com essas operações.
- Projeto de mensagem no qual se projeta a estrutura das mensagens enviadas e recebidas pelo serviço.
- Desenvolvimento WSDL no qual se traduz o projeto lógico e de mensagem para uma descrição abstrata de interface escrita em WSDL.

O primeiro estágio – projeto lógico de interface – começa com os requisitos de serviço e são definidos os parâmetros e nomes das operação. Neste estágio pode-se definir as exceções que podem surgir quando uma operação de serviço é chamada.

4.3 - IMPLEMENTAÇÃO E IMPLANTAÇÃO DE SERVIÇO

Uma vez que foram identificados os serviços candidatos e a interface foi projetada, o estágio final do processo de engenharia de serviços é a implementação do serviço. Esta etapa envolve a programação dos serviços usando uma linguagem padronizada de programação como Java ou C#, que possuem bibliotecas com apoio extensivo para desenvolvimento de serviços.

Na medida em que o serviço tenha sido implementado, ele deve então ser testado antes que seja implantado. Isso envolve o exame e o particionamento das entradas de serviços, a criação de mensagens de entrada que refletem essas combinações de entradas, e então, a verificação das saídas esperadas.

A implantação do serviço, o estágio final do processo, envolve tornar o serviço disponível para o uso no servidor Web. Se a intenção for disponibilizar o serviço publicamente, será necessário escrever uma descrição UDDI para que os usuários potenciais possam achar o serviço.

CAPÍTULO 5 – ESTUDO DE CASO

O objetivo deste capítulo é apresentar um estudo de caso realizado com base no processo de desenvolvimento descrito no capítulo anterior para criar um serviço de distribuição de notícias *online*. A figura 15 apresenta os três estágios do processo. Consideramos para realizar o estudo de caso, que uma grande empresa jornalística que atua no setor de comunicações, mais especificamente na produção e veiculação de notícias, decidiu diversificar a distribuição de notícias e atuar também em uma das principais e atuais mídias, a internet, através da publicação diária de conteúdo jornalístico.

Figura 15: Processo de desenvolvimento
Elaboração própria baseada em Sommerville (2011)

O processo de desenvolvimento sofreu algumas adaptações, descritas a seguir:

- ✓ No projeto de serviço foram incluídos modelos elaborados para que houvesse melhor compreensão das classes criadas.
- ✓ A apresentação do WSDL foi transferida do projeto de serviço para seção de implementação e implantação de serviço considerando que nesse projeto o WSDL não foi gerado antes da codificação. No desenvolvimento deste serviço o WSDL foi gerado a partir do código elaborado.
- ✓ O projeto de interface de serviço foi composto apenas pelo estágio de projeto lógico de interface.

As próximas seções apresentam os três estágios do processo de desenvolvimento:

- Identificação de serviço candidato
- Projeto de serviço
- Implementação e implantação de serviço

5.1 - IDENTIFICAÇÃO DE SERVIÇO CANDIDATO

Nesta seção será apresentada a identificação do serviço candidato e a definição dos requisitos de serviço.

Um Web Service baseado em SOAP será desenvolvido com o objetivo de disponibilizar notícias. Este serviço estará acessível para localização e utilização por aplicações clientes.

Um exemplo de como as informações geradas pelo Web Service podem ser utilizadas por uma aplicação cliente pode ser observado na figura 16.

Figura 16: Página web do site Destakes

O destakes é um site informativo que divulga notícias. As notícias estão agrupadas por categorias. Exemplos de categorias são Tecnologia e Negócios. Além da categoria, cada notícia possui um título, uma descrição, uma fonte e o tempo decorrido a partir do cadastramento da notícia.

Este site serviu de inspiração para a construção do Web Service. Os atributos das notícias que foram considerados para a elaboração do Web Service, foram definidos a partir

do modelo deste site, mas a fonte não foi incluída, pois as informações foram cadastradas manualmente no banco de dados local.

O serviço é identificado como Serviço utilitário, pois implementa uma funcionalidade geral, que pode ser usada por diferentes processos de negócio.

- Requisitos Funcionais

- O serviço deve fornecer uma lista de notícias de uma categoria em um determinado intervalo de tempo, indicado por uma data de início e uma data de fim. A lista deve conter o título, o texto da notícia e a data de cadastro.
- O serviço deve disponibilizar os nomes das categorias de notícias que estão disponíveis.

- Requisitos não funcionais

- O serviço estará disponível sem interrupção durante 24horas.
- As informações necessárias para consumo do serviço estarão disponíveis para os clientes.
- O serviço é fornecido a qualquer cliente sem necessidade de cadastramento e sem custos.

5.2 - PROJETO DE SERVIÇO

No projeto lógico de interface são identificadas as operações associadas ao serviço, suas entradas, saídas e as exceções associadas a estas operações. A tabela 1 mostra as operações que implementam os requisitos.

Operação	Descrição
getNoticias	Retorna uma lista de notícias de uma categoria em um determinado intervalo de tempo, indicado por uma data de início e uma data de fim. A lista deve conter o título, o texto da notícia e a data de cadastro.
getCategorias	Retorna uma lista de categorias.

Tabela 1: Descrição informal das operações do serviço

A tabela 2 ilustra a estrutura do projeto de interface com as entradas e saídas das operações do Serviço e sua respectiva exceção.

Operação	Entradas	Saídas	Exceção
getNoticias	**gNIn** Nome da categoria Data inicial Data final	**gNOut** Lista de notícias contendo: Título, texto e data de cadastro	**ObjetoNaoEncontradoException** Categoria não possui notícias no período indicado
getCategorias	----	**gCOut** Lista de nomes de categorias	----

Tabela 2: Projeto de Interface

5.3 - IMPLEMENTAÇÃO E IMPLANTAÇÃO DE SERVIÇO

Depois de ter identificado o serviço candidato e projetado suas interfaces, o estágio final do processo de engenharia de serviço é a implementação e implantação do serviço. Esta etapa está dividida em quatro estágios, descritos a seguir: Modelagem, codificação, testes e implantação.

(1° Estágio) – Modelagem

- Modelagem para gerar as tabelas Categoria e Noticia

A seguir a figura 17, ilustra o esquema do banco de dados relacional, elaborado com auxílio da ferramenta Case Studio 2.

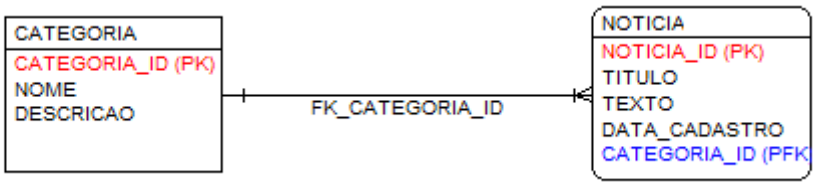


Figura 17: Diagrama para o esquema do banco de dados relacional

- Modelagem para criação das classes do Web Service

A figura 18, ilustra o diagrama de classe de projeto do serviço.

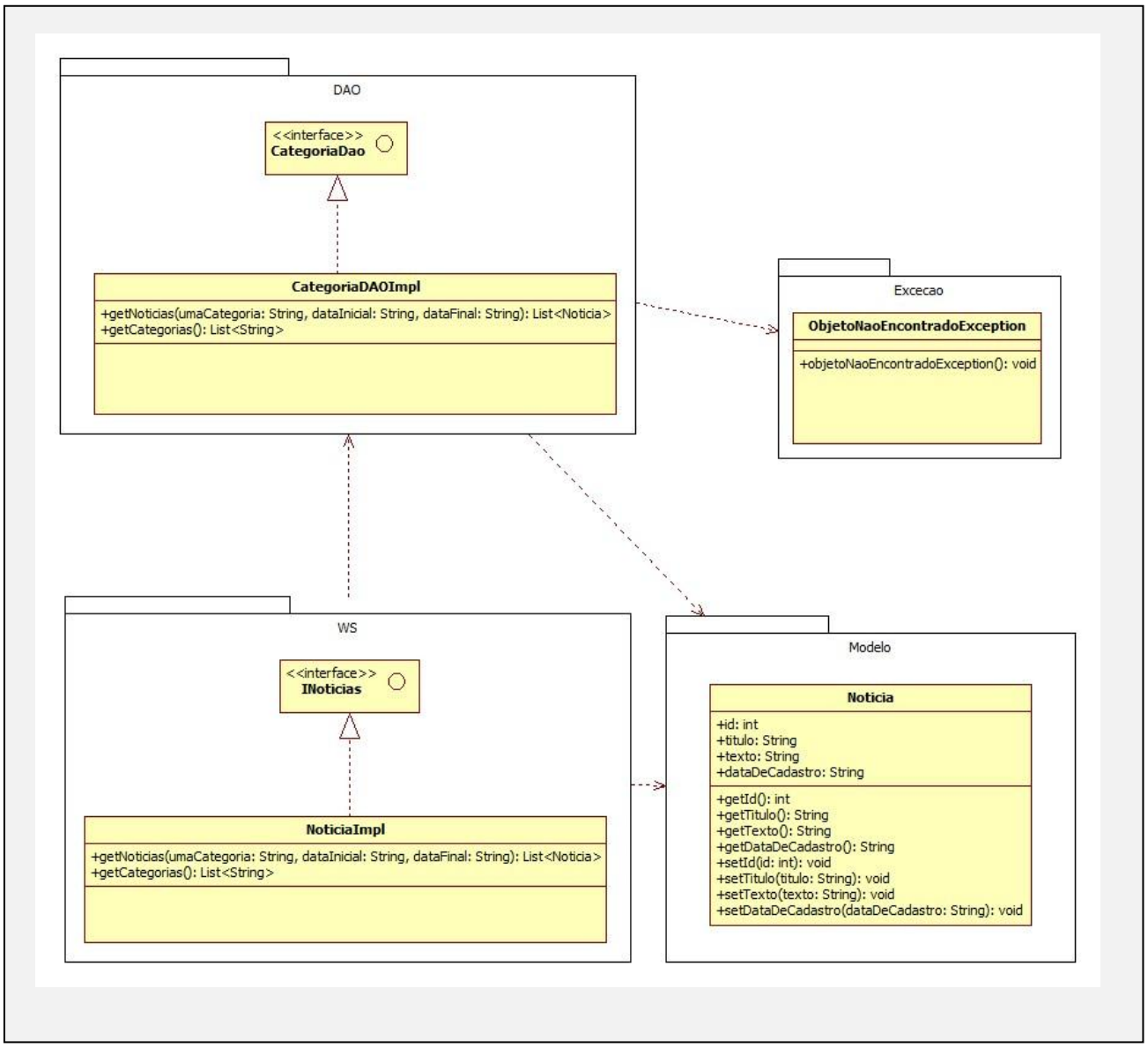


Figura 18: Diagrama de classe

- Descrição dos métodos

Pacote DAO (*Data Access Object*)

getCategorias -> Retorna uma lista de nomes de categorias a partir de uma busca no banco de dados.

getNoticias -> Faz a busca no banco de dados, cria as noticias, insere na lista e retorna a lista.

Pacote WS

Foi seguida a orientação apresentada em Kalim (2009), que recomenda a elaboração de uma interface que declare os métodos do Web Service e uma implementação que define os métodos declarados na interface.

getCategorias -> Chama o método getCategorias de CategoriaDAO, que retorna uma lista de nomes de categoria.

getNoticias -> Chama o método getNoticias de CategoriaDAO, que retorna uma lista de noticias de uma categoria no período indicado.

(2° Estágio) – Codificação

O Web Service WSNOTICIAS, foi desenvolvido sob o sistema operacional Windows 7 Home Premium, utilizando o Eclipse, a linguagem de programação Java e banco de dados Mysql Server versão 5.1.50.

A organização W3C (World Wide Web Consortium) define alguns padrões para o funcionamento de um Web Service. Em geral, as plataformas de maior uso comercial implementam a arquitetura definida pelos padrões da W3C.

Na plataforma Java, há especificações que definem a implementação Java dos padrões estabelecidos pelo W3C. A especificação Java diretamente relacionada a WebServices que seguem os padrões da W3C é a Java API for XML-Based Web Services - JAX-WS.

Segundo Abinader (2006), todas as bibliotecas exigidas para compilar, executar e consumir WebService estão disponíveis no Java 6, que suporta JAX-WS. Este suporta serviços baseados em SOA.

JAX-WS é comumente abreviado para JWS de Java WebServices. A figura19 demonstra como é realizada a requisição a um método de um WebService sobre a plataforma Java usando o pacote JAX-WS.

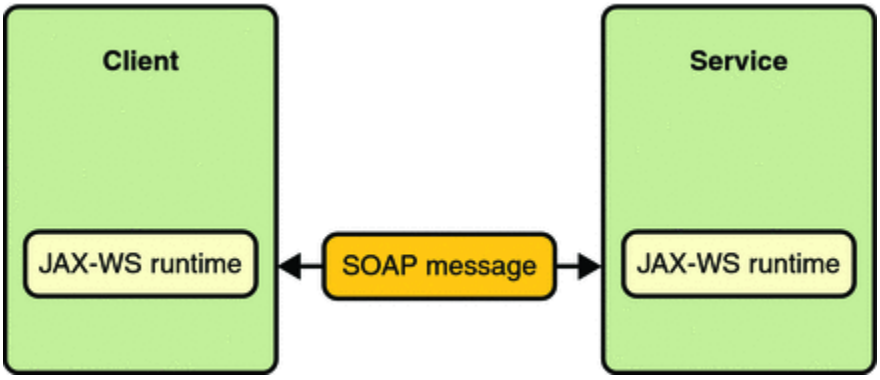


Figura 19: Comunicação entre um WebService (com JAX-WS) e um Cliente
Fonte: Oracle (2011)

No lado cliente, há apenas a necessidade de um computador que tenha instalado em seu sistema, um navegador (browser) de internet. No lado servidor, há a necessidade de estar devidamente instalado o Tomcat versão 7.0 (servidor Web). Neste caso o servidor deve ter instalado previamente o Ambiente de Execução Java (JRE versão 1.6), o Java API for XML-Based Web Services - JAX-WS e o Mysql 5.1.

As figuras 20 e 21 apresentam o código da interface INoticias e da classe de implementação NoticiaImpl.

Classe INoticias

```
@WebService
public interface INoticias {

    @WebMethod(operationName = "getNoticias")
    public List<Noticia> getNoticias(
            @WebParam(name = "umaCategoria") String umaCategoria,
            @WebParam(name = "dataInicial") String dataInicial,
            @WebParam(name = "dataFinal") String dataFinal) throws ObjetoNaoEncontradoException;

    @WebMethod(operationName = "getCategorias")
    public List<String> getCategorias(); // retorna o nome das categorias

}
```

Figura 20: Interface INoticias

Classe NoticiaImpl

```
@WebService(endpointInterface = "ws.INoticias")
public class NoticiaImpl implements INoticias {

    @Override
    public List<Noticia> getNoticias(String umaCategoria, String dataInicial, String dataFinal) throws ObjetoNaoEncontradoException
    {
        CategoriaDAO categoriaDAO = new CategoriaDAOImpl(); // criação de uma instância de CategoriaDAO
        // categoriaDAO.getNoticias() Faz a busca no banco de dados, cria as notícias, insere na lista e retorna a lista.
        // Neste caso o retorno do método é armazenado na variável noticias.
        List<Noticia> noticias = categoriaDAO.getNoticias(umaCategoria, dataInicial, dataFinal);
        // se o tamanho da lista de notícias for menor que 1 lança uma exceção
        if (noticias.size() < 1) {
            throw new ObjetoNaoEncontradoException("Categoria não possui notícias no período indicado.");
        }
        return noticias;
    }

    @Override
    public List<String> getCategorias()
    {
        CategoriaDAO categoriaDAO = new CategoriaDAOImpl(); // criação de uma instância de CategoriaDAO
        // Retorna uma lista de nomes de categorias do banco de dados.
        return categoriaDAO.getCategorias();

    }
}
```

Figura 21: Classe NoticiaImpl

A interface é chamada INoticias e a implementação é chamada NoticiasImpl. A interface deve ter a anotação **@WebService** para indicar que é a interface de um Web Service

e na classe de implementação além desta anotação deve conter o parâmetro endpointInterface que liga a interface INoticias a classe NoticiaImpl.

Neste projeto o WebService Notícias (WSNoticias) tem a função de executar uma rotina de exibição de notícias, através dos métodos implementados. Cada método possui a anotação **@WebMethod** para que possa ser reconhecido como uma operação do Web Service WSNoticias e ser referenciado por outras aplicações quando o serviço for publicado.

(3° e 4° Estágios) – Testes e implantação do Web Service WSNOTICIAS

Um documento WSDL é um contrato entre o serviço e seus clientes. O contrato disponibiliza informações essenciais como: o endpoint do serviço, as operações de serviço e os tipos de dados exigidos para estas operações. O WSDL gerado pelo serviço WSNoticias está descrito no apêndice A.

Na Figura 22 é apresentado parte da classe Endpoint que fará a publicação do Web Services WSNoticias.

```
// primeiro argumento é a URL de publicação
// segundo argumento é uma instância da implementação do Web service
Endpoint.publish("http://127.0.0.1:9876/wsnoticias", new NoticiaImpl());
```

Figura 22: Publicação do WSNoticias

Os testes foram realizados a partir da ferramenta Eclipse, que após compilar o projeto, executa o mesmo exibindo uma janela do navegador (Figura 23) com os dois métodos implementados, após inicializar o servidor web Apache Tomcat.

Ao clicar no link do método getNoticias uma nova página no navegador é exibida, conforme demonstra a Figura 24.

Na Figura 24 são exibidos: a) campos para entrada dos valores para testar o método; b) o elemento envelope da mensagem SOAP que será enviado via HTTP para o Web Service.

Informando valores nos campos mostrados no formulário acima e pressionando o botão *Go*, uma nova tela é exibida conforme demonstra a Figura 25.

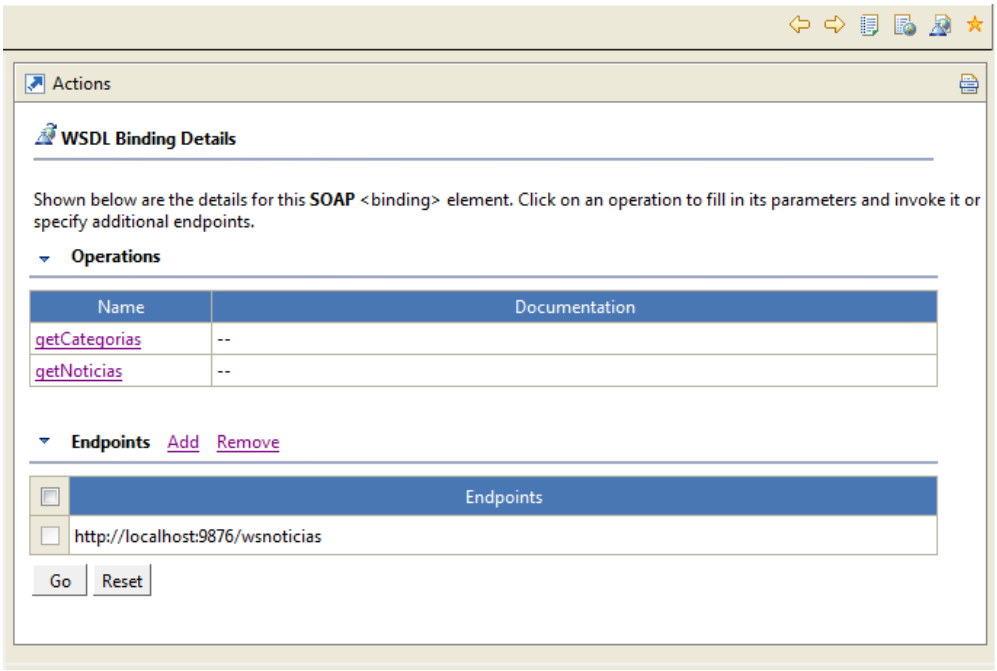


Figura 23: Página gerada pelo eclipse para teste do Web Service

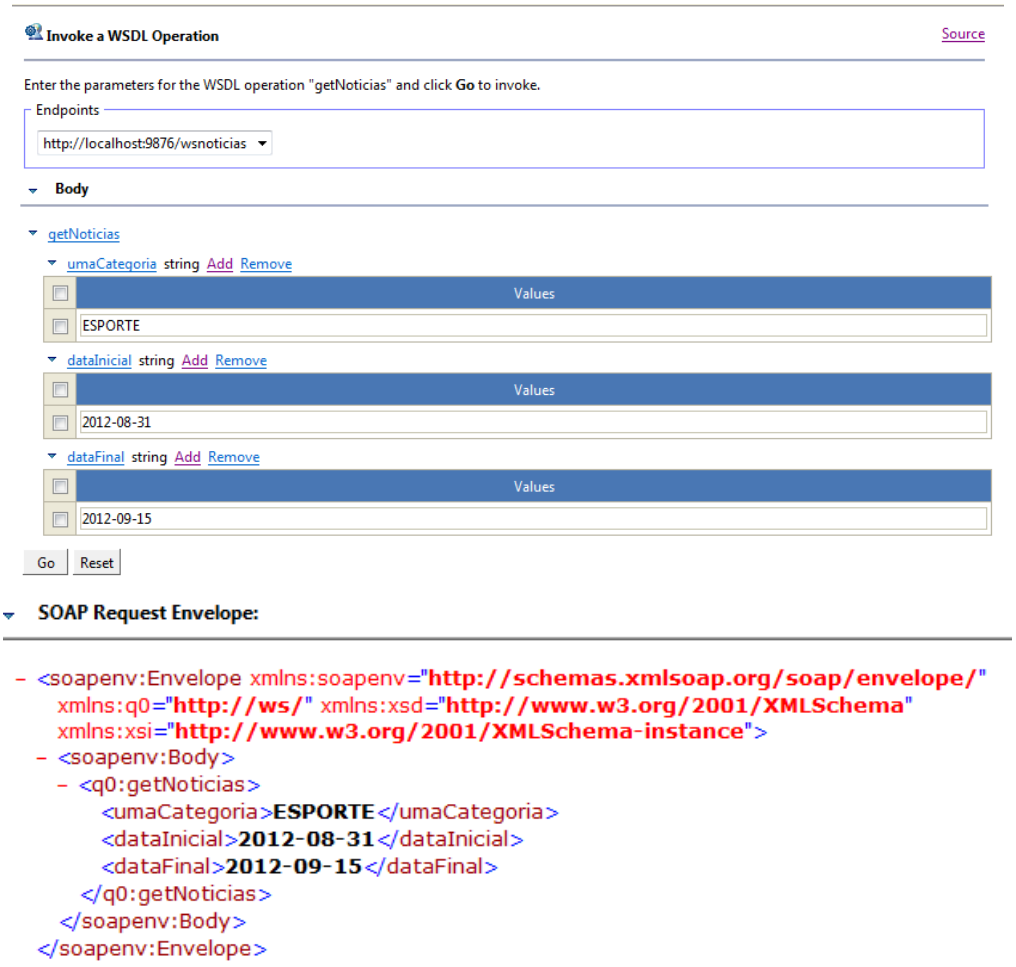


Figura 24: Página gerada pelo Eclipse para teste do Web Service (operação getNoticias)

Figura 25: Página gerada pelo Eclipse para teste do Web Service (operação getNoticias)

Na página mostrada na figura 25 podem ser observados os dados que retornam do método getNoticias do WSNoticias, como também o elemento envelope da mensagem SOAP que será recebido pelo cliente via HTTP.

Foi realizado também um teste elaborando-se uma aplicação consumidora do Web Service. A figura 26 apresenta o fluxo do processo realizado.

O Web Service publica o documento WSDL e a partir de então, o cliente tem acesso e as informações necessárias ao consumo.

A aplicação cliente envia uma solicitação, utilizando o protocolo de transporte HTTP, ao Web Service . O Web Service acessa a base de dados e coleta as informações solicitadas e envia a resposta ao cliente.

A aplicação cliente fará a requisição ao Web Service WSNoticias, passando pelos dois métodos implementados. A figura 27 mostra o método elaborado que faz parte do processo de comunicação com o Web Service.

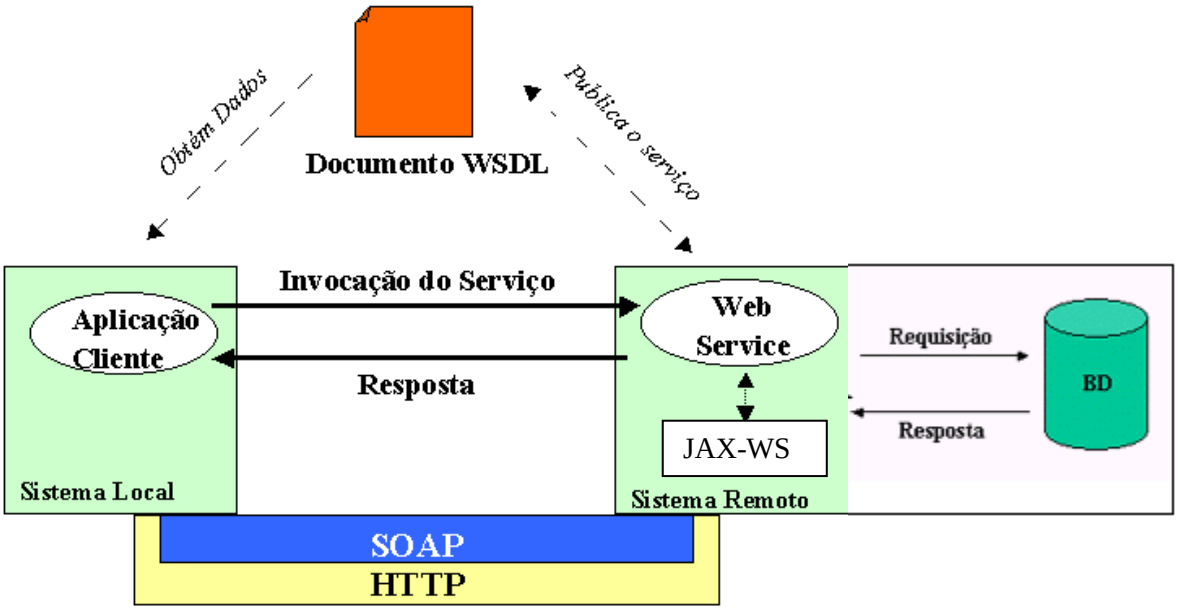


Figura 26: Fluxo de processos

```
public INoticias acessarInterfaceWS() throws Exception{
    // URL fornece o endpoint, no qual o serviço pode ser acessado
    URL url = new URL("http://localhost:9876/wsnoticias?wsdl");

    // QName é um nome qualificado XML, que consiste do nome local do serviço (neste caso NoticiaImplService)
    // Primeiro argumento é o URI do serviço
    // Segundo é o nome do serviço publicado no WSDL
    // cria a comunicação com o Web Service
    QName qname = new QName("http://ws/", "NoticiaImplService");

    // Cria, de fato, um fábrica para o serviço
    Service service = Service.create(url, qname);

    // recupera uma classe que implemente a interface do Web Service
    INoticias iNoticias = service.getPort(INoticias.class);

    return iNoticias;

}
```

Figura 27: Método acessarInterfaceWS

O cliente chama o método Service.create com dois argumentos: uma URL que fornece o *endpoint*, no qual o serviço pode ser acessado, e um nome qualificado XML (um QName Java), que, por sua vez, consiste do nome local do serviço (neste caso, NoticiaImplService) e um identificador de namespace (neste caso, http://ws/).

Depois dos objetos URL e QName serem construídos e do método Service.create ser chamado, a expressão que interessa:

INoticias iNoticias = service.getPort(INoticias.class);

O cliente então poderá utilizar o objeto iNoticias para chamar os dois métodos do Web Service:

- iNoticias.getCategorias() – para obter a lista de nomes de categorias.

- iNoticias.getNoticias() – para obter a lista de notícias.

O cliente Java chama os dois métodos do Web Service; e as bibliotecas Java geram e processam as mensagens SOAP trocadas transparentemente, para possibilitar as solicitações com sucesso.

Na Figura 28 é apresentada a tela da aplicação, sendo ela específica para testar os métodos do Web Services.

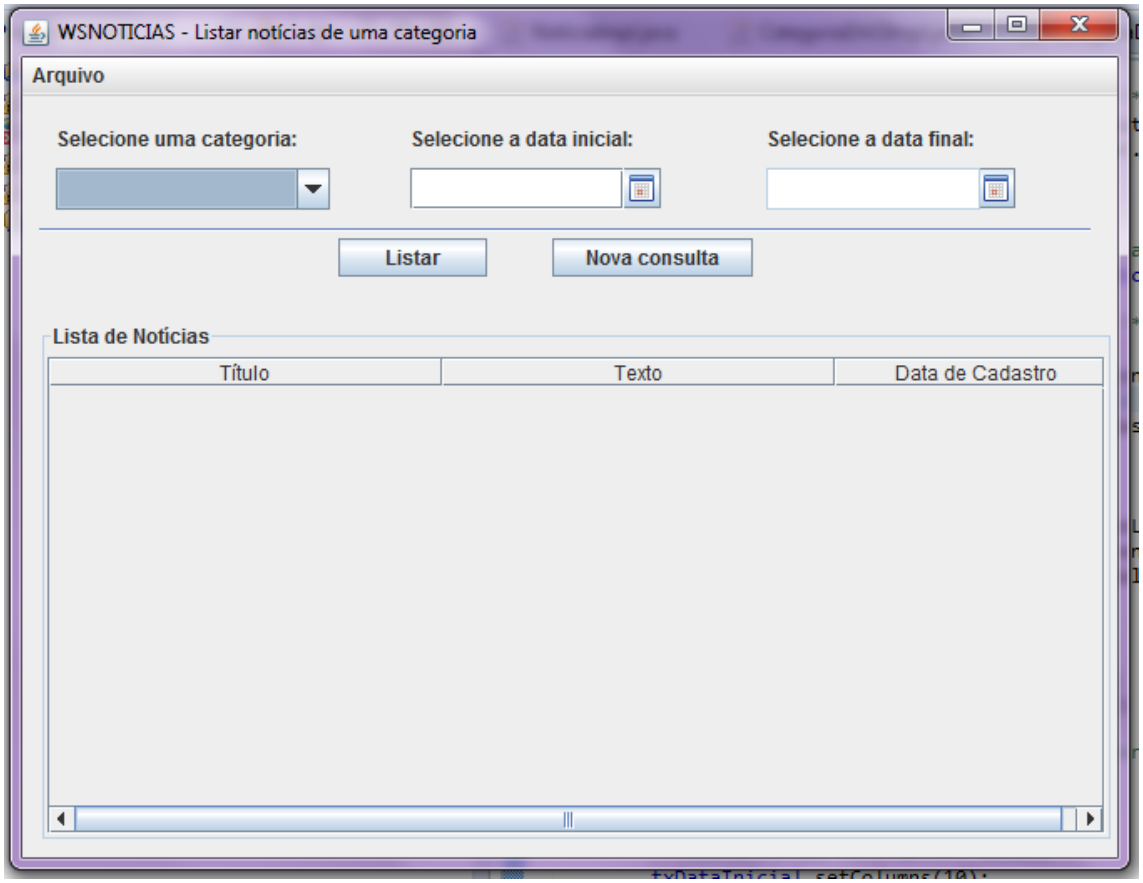


Figura 28: Tela inicial da aplicação cliente

Na figura 29, é apresentada a tela que irá efetuar a requisição ao método getCategorias para exibir as categorias disponíveis.

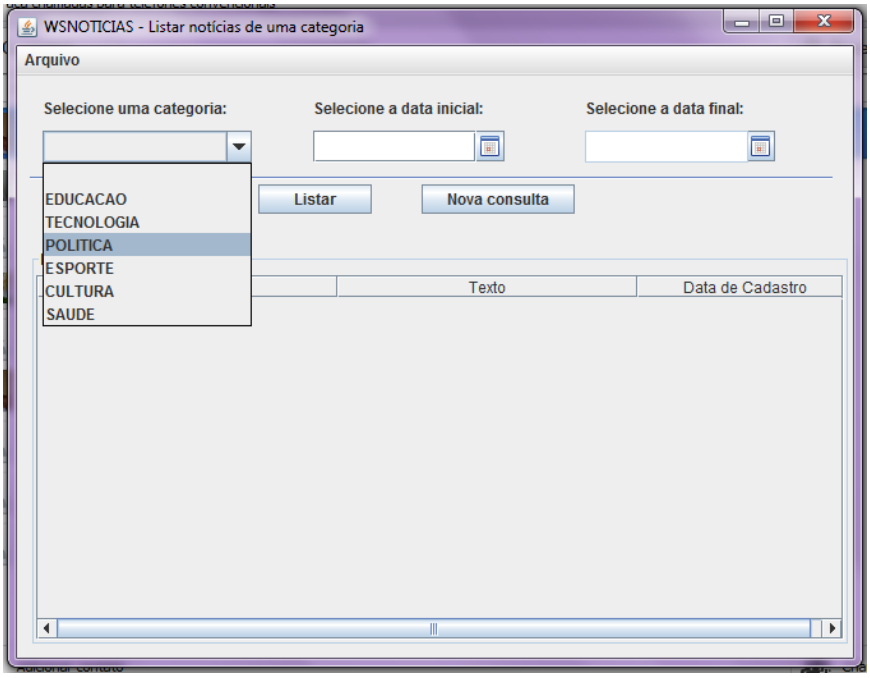


Figura 29: Tela da aplicação cliente durante a seleção da categoria

Para que sejam listadas as notícias deve-se selecionar a categoria e preencher o campo data inicial e data final como mostra a figura 30.

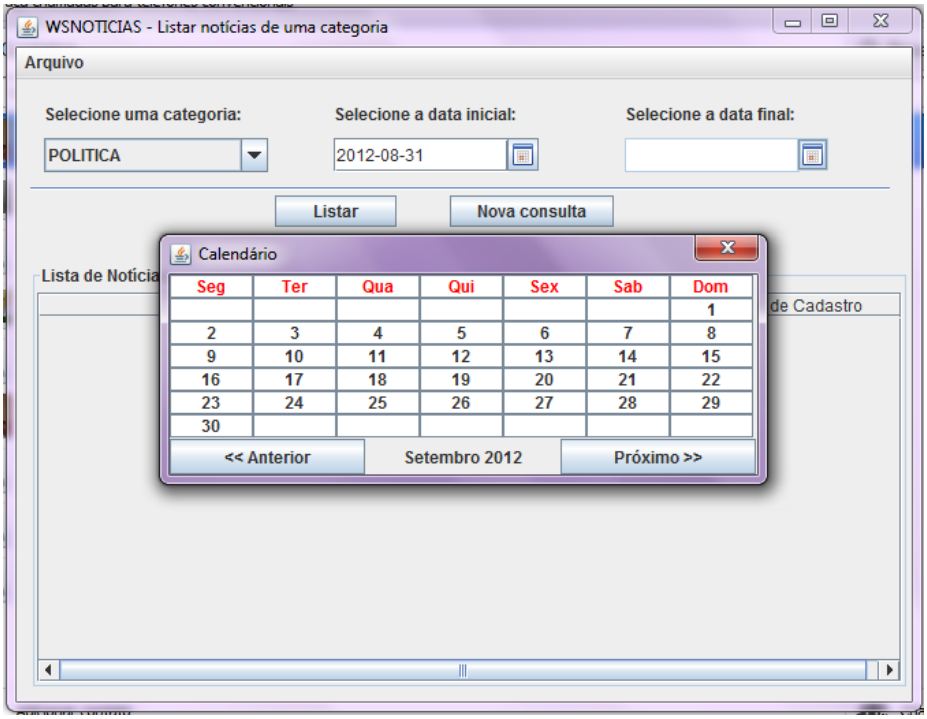


Figura 30: Tela da aplicação cliente durante inclusão das datas

Após selecionar uma categoria, data inicial e final, já temos os parâmetros necessários para o serviço executar a consulta.

Quando o botão Listar é pressionado, o método getNoticias é executado e suas informações são apresentadas conforme apresentado na figura 31.

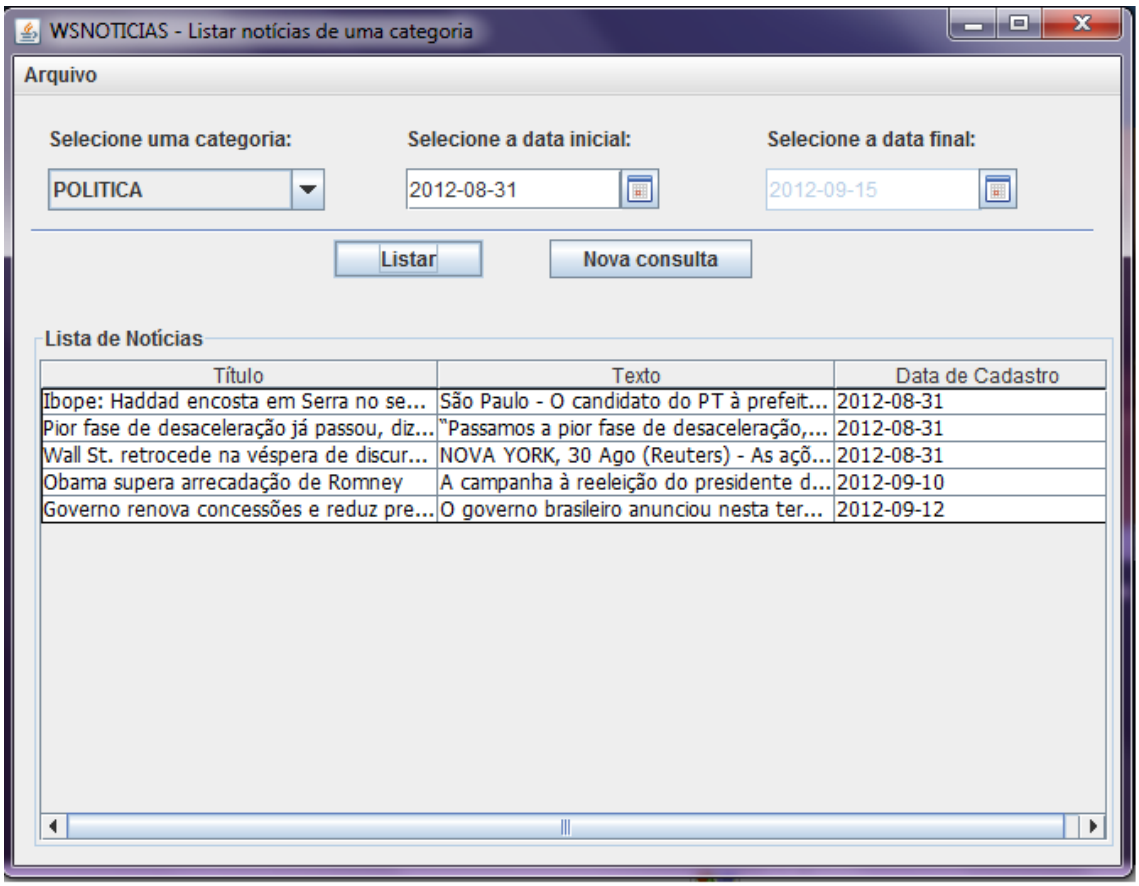


Figura 31: Tela da aplicação cliente após o retorno da lista de notícias

CAPÍTULO 6 - CONCLUSÃO

Este trabalho teve como objetivo principal o desenvolvimento de um estudo de caso aplicando conceitos da arquitetura SOA, Web Services e processos de desenvolvimento de serviços para reúso. Possibilitou o enriquecimento e adicionou conhecimento em diversos assuntos, dentre eles: (i) linguagem de programação Java; (ii) arquitetura orientada a serviços e web service; (iii) processo de desenvolvimento de serviços para reúso e (iv) banco de dados MySQL.

Os requisitos determinados para o Web Service foram cumpridos quase na sua totalidade. Então, podemos dizer que o nosso objetivo foi alcançado, ainda que o mesmo ainda não tenha sido publicado no repositório UDDI.

Vimos que os Web Services podem ser organizados em dois grupos: SOAP e REST. Neste trabalho foi abordado o desenvolvimento de um Web Service baseado em SOAP. Atualmente além de serem elaborados Web Services SOAP, há um grande enfoque no estilo REST. Assim, trabalhos posteriores poderão se basear no desenvolvimento de Web Services REST realizando uma comparação entre os dois estilos de desenvolvimento de Web Services.

REFERÊNCIAS BIBLIOGRÁFICAS

ABINADER, Jorge Abilio. **Web Services em Java**. 1 ed. São Paulo: Brasport, 2006.

ANDERSON, Richard et al. **Professional XML**. 1. ed. Rio de Janeiro: Ciência Moderna, 2001.

COULOURIS, G.; DOLLIMORE, J.; KINDBERG, T. **Sistemas Distribuídos - Conceitos e Projeto**. Brasil: Bookman Companhia Ed, 2010.

COUTINHO, J. **Um Estudo sobre o Desenvolvimento Orientado a Serviços**. Dissertação de Mestrado. PUC, 2004.

ERL, T. **Service Oriented Architecture: Concepts, Technology and Design**. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2005.

JOSUTTIS, N. M.. **SOA in Practice – The Art of Distributed System Design**. 1. ed. O"Reilly, 2007.

KALIM, Martin. **Java Web Services: Implementando**. 1. ed. Rio Janeiro: Alta Books, 2009.

KUMAR, B.V., Prakash Narayan, Tony Ng. **Implementando SOA usando Java EE.** Rio de Janeiro: Alta Books, 2012.

MCGOVEN, Tyagi, Stevens, Mathew. **Java Web Services Architecture.** Morgan Kaufmann, 2003.

MOULTIS, Natanya Pitts; KIRK, Cheryl. **XML: black book**. 1. ed. São Paulo: MAKRON Books, 2000.

SOMMERVILLE, Ian. **Engenharia de software**, 9 edição. São Paulo: Pearson Addison, 2011.

W3C. World Wide Web Consortium 2009. Disponível em: <http://www.w3.org/TR/2000/NOTE-SOAP-20000508>. Acessado em: 2 Julho de 2012.

W3C. Web Services Architecture: W3C Working Group 2004a. Disponível em: <http://www.w3.org/TR/ws-arch/ >. Acessado em: 15 agosto 2012.

ORACLE. The Java EE 5 Tutorial. Disponível em: <http://docs.oracle.com/javaee/5/tutorial/doc/bnayn.html>. Acessado em: 05/05/2012.

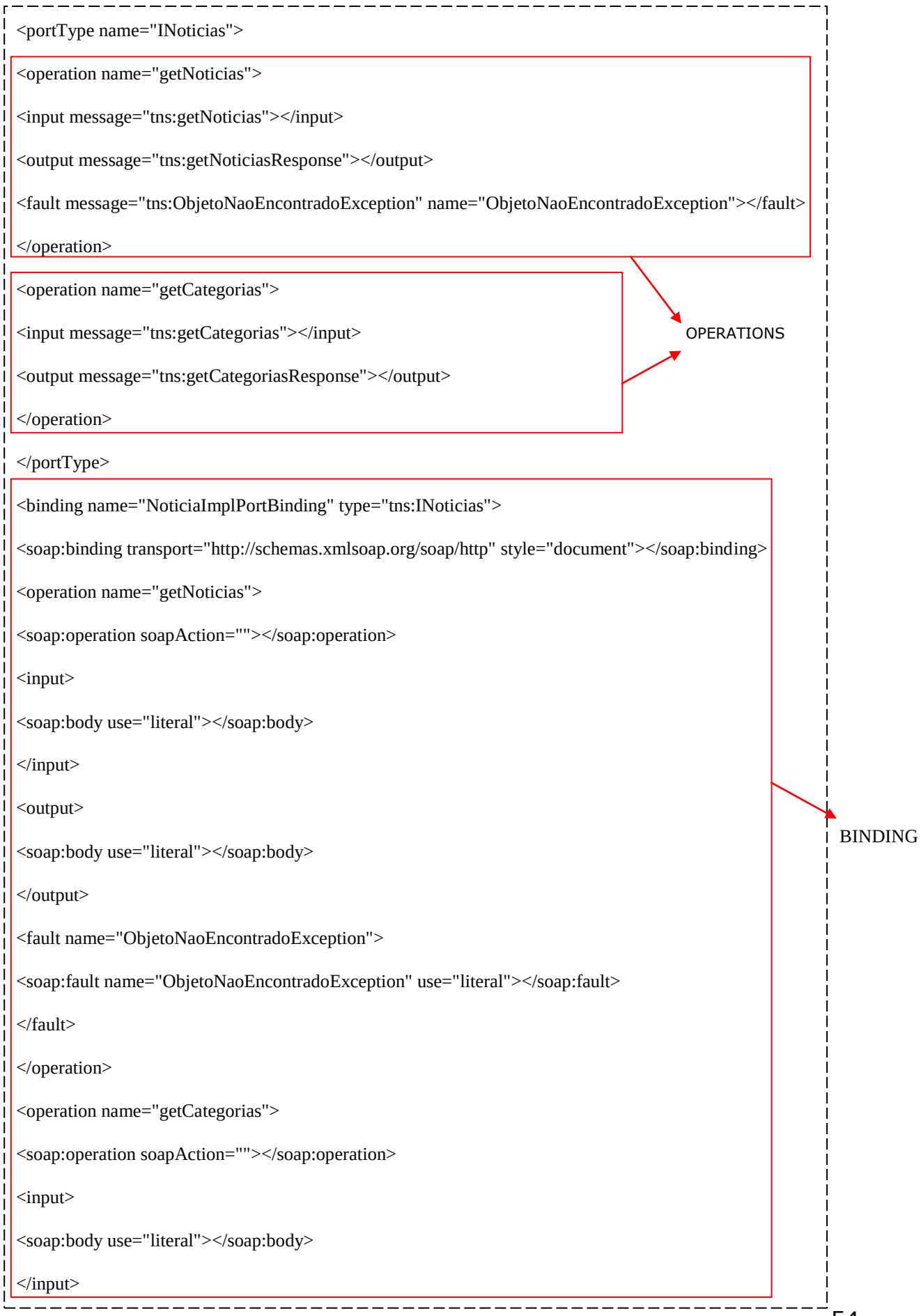
APÊNDICE A

I. WSDL gerado pelo código

```
<?xml version="1.0" encoding="UTF-8"?>

<definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:tns="http://ws/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="http://ws/" name="NoticiaImplService">

<types>
<xsd:schema>
<xsd:import namespace="http://ws/" schemaLocation="http://localhost:9876/wsnoticias?xsd=1"></xsd:import>
</xsd:schema>
</types>

<message name="getNoticias">
<part name="parameters" element="tns:getNoticias"></part>
</message>
<message name="getNoticiasResponse">
<part name="parameters" element="tns:getNoticiasResponse"></part>
</message>
<message name="ObjetoNaoEncontradoException">
<part name="fault" element="tns:ObjetoNaoEncontradoException"></part>
</message>
<message name="getCategorias">
<part name="parameters" element="tns:getCategorias"></part>
</message>
<message name="getCategoriasResponse">
<part name="parameters" element="tns:getCategoriasResponse"></part>
</message>
```

TYPE

MESSAGES


<portType name="INoticias">
<operation name="getNoticias">
<input message="tns:getNoticias"></input>
<output message="tns:getNoticiasResponse"></output>
<fault message="tns:ObjetoNaoEncontradoException" name="ObjetoNaoEncontradoException"></fault>
</operation>
<operation name="getCategorias">
<input message="tns:getCategorias"></input>
<output message="tns:getCategoriasResponse"></output>
</operation>
OPERATIONS
</portType>
<binding name="NoticiaImplPortBinding" type="tns:INoticias">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"></soap:binding>
<operation name="getNoticias">
<soap:operation soapAction=""></soap:operation>
<input>
<soap:body use="literal"></soap:body>
</input>
<output>
<soap:body use="literal"></soap:body>
</output>
<fault name="ObjetoNaoEncontradoException">
<soap:fault name="ObjetoNaoEncontradoException" use="literal"></soap:fault>
</fault>
</operation>
<operation name="getCategorias">
<soap:operation soapAction=""></soap:operation>
<input>
<soap:body use="literal"></soap:body>
</input>
BINDING

```
<output>

<soap:body use="literal"></soap:body>

</output>

</operation>

</binding>

<service name="NoticiaImplService">

<port name="NoticiaImplPort" binding="tns:NoticiaImplPortBinding">

<soap:address location="http://localhost:9876/wsnoticias"></soap:address>

</port>

</service>

</definitions>
```

II. XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:tns="http://ws/" xmlns:xs="http://www.w3.org/2001/XMLSchema" version="1.0"
targetNamespace="http://ws/">

<xs:element name="ObjetoNaoEncontradoException"
type="tns:ObjetoNaoEncontradoException"></xs:element>

<xs:element name="getCategorias" type="tns:getCategorias"></xs:element>

<xs:element name="getCategoriasResponse" type="tns:getCategoriasResponse"></xs:element>

<xs:element name="getNoticias" type="tns:getNoticias"></xs:element>

<xs:element name="getNoticiasResponse" type="tns:getNoticiasResponse"></xs:element>

<xs:complexType name="getCategorias">
<xs:sequence></xs:sequence>
</xs:complexType>

<xs:complexType name="getCategoriasResponse">
<xs:sequence>
<xs:element name="return" type="xs:string" minOccurs="0" maxOccurs="unbounded"></xs:element>
</xs:sequence>
</xs:complexType>

<xs:complexType name="getNoticias">
<xs:sequence>
<xs:element name="umaCategoria" type="xs:string" minOccurs="0"></xs:element>
<xs:element name="dataInicial" type="xs:string" minOccurs="0"></xs:element>
<xs:element name="dataFinal" type="xs:string" minOccurs="0"></xs:element>
</xs:sequence>
</xs:complexType>
```

```
<xs:complexType name="getNoticiasResponse">
<xs:sequence>
<xs:element name="return" type="tns:noticia" minOccurs="0" maxOccurs="unbounded"></xs:element>
</xs:sequence>
</xs:complexType>

<xs:complexType name="noticia">
<xs:sequence>
<xs:element name="dataDeCadastro" type="xs:string" minOccurs="0"></xs:element>
<xs:element name="id" type="xs:int"></xs:element>
<xs:element name="texto" type="xs:string" minOccurs="0"></xs:element>
<xs:element name="titulo" type="xs:string" minOccurs="0"></xs:element>
</xs:sequence>
</xs:complexType>

<xs:complexType name="ObjetoNaoEncontradoException">
<xs:sequence>
<xs:element name="message" type="xs:string" minOccurs="0"></xs:element>
</xs:sequence>
</xs:complexType>

</xs:schema>
```