FUNDAÇÃO DE ENSINO EURÍPIDES SOARES DA ROCHA
CENTRO UNIVERSITÁRIO "EURÍPIDES DE MARÍLIA" – UNIVEM
CURSO DE CIÊNCIA DA COMPUTAÇÃO

ANDERSON LIBERATO DA SILVA

IMPLEMENTAÇÃO DE "WEB SERVICE" EM PLATAFORMA JAVA – LADO SERVIDOR

MARÍLIA
2005

ANDERSON LIBERATO DA SILVA

IMPLEMENTAÇÃO DE "WEB SERVICE" EM PLATAFORMA JAVA - LADO SERVIDOR

Monografia apresentado ao Curso de Bacharelado em Ciência da Computação do Centro Universitário Eurípides de Marília, mantido pela Fundação de Ensino Eurípides Soares da Rocha, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador:
Prof. Ms. Ricardo Petruzza do Prado

MARÍLIA
2005

SILVA, Anderson Liberato da.
Implementação de “Web Service” em Plataforma Java – Lado Servidor / Anderson Liberato da Silva; orientador: Ricardo Petruzza do Prado.
Marília, SP: [s,n], 2005.

Monografia (Bacharelado em Ciência da Computação) – Centro Universitário Eurípides de Marília – Fundação de Ensino Eurípides Soares da Rocha.

1. Serviços Web 2. Linguagem XML 3. Plataforma J2EE

CDD: 005.1151

ANDERSON LIBERATO DA SILVA

IMPLEMENTAÇÃO DE "WEB SERVICE" EM PLATAFORMA JAVA – LADO SERVIDOR

Banca examinadora da monografia apresentado ao Curso de Bacharelado em Ciência da Computação da UNIVEM,/F.E.E.S.R., como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Resultado: _____.

ORIENTADOR: Prof. Ms. Ricardo Petruzza do Prado

1º EXAMINADOR: Prof. Dr. Marcos L. Mucheroni

2º EXAMINADOR: Profª Drª Maria Istela Cagnin

Marília, 24 de Novembro de 2005.

SILVA, Anderson Liberato da. **Implementação de "Web Service" em Plataforma Java – Lado Servidor.** 2005. Monografia (Bacharelado em Ciência da Computação) – Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2005.

RESUMO

Com a necessidade cada vez maior das empresas integrarem dados e aplicações, tanto para possibilitar a troca de informações entre seus sistemas internos, como para com seus fornecedores, surge a necessidade de um padrão que torne este intercâmbio possível. Utilizando-se da Internet como meio de trocar estas informações foram criados os Serviços Web. Os Serviços Web oferecem um meio de interoperabilidade entre aplicações, sendo possíveis suas execuções em diversas plataformas. Neste trabalho foram estudas as principais tecnologias que compõem a infra-estrutura de Serviços Web, tais como, Linguagem XML, Protocolo SOAP, Especificação WSDL, Padrão UDDI entre outras. Utilizou-se a plataforma J2EE que oferece suporte ao desenvolvimento de Serviços Web através da linguagem de programação Java. Com base no estudo das tecnologias envolvidas na infra-estrutura de Serviços Web, e também, das tecnologias oferecidas pela plataforma J2EE, foi criado o Serviço Web ProntuarioWS, somente o lado do servidor, executando sobre o sistema operacional Linux. Este serviço oferece operações para centralização e recuperação de informações sobre prontuários de pacientes de Instituições de Saúde.

Palavras-chave: Serviços Web. Linguagem XML. Plataforma J2EE.

SILVA, Anderson Liberato da. **Implementação de "Web Service" em Plataforma Java – Lado Servidor.** 2005. Monografia (Bacharelado em Ciência da Computação) – Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2005.

ABSTRACT

With the increasing needed for the corporations to integrate data and applications, to make possible the change of information between their internal systems, such for their suppliers, rise the needed for a standard, which make possible these interchange. Using the Internet like the way to change this information, were created the Web Services. The Web Services offers a way to interoperate applications, been possible their executions in different platforms. In this work were studied the main technologies who are the basis of the Web Services, like XML Language, SOAP Protocol, WSDL specifications, the UDDI pattern and others. The platform J2EE was used because offers support to development of Web Services using the Programming Language Java. With base in the study of the technologies involved in the basis of Web Service, and the technologies offers by the J2EE platform, was creates the Web PromptuaryWS Service, only server side, running over Linux Operating System. This service offers operations for the centralization e recovery of the information about patient's promptuary from the Health Institutions.

Keywords: Web Service**.** XML Language. J2EE platform.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

B2B: *Business to Business - Operações realizadas entre empresas.*

DCOM: *Distributed Component Object Model – Modelo de Componente Distribuído.*

DOM: *Document Object Model – Modelo de Objeto de Documento.*

DTD: *Document Type Definition – Definição de Tipos de Documentos.*

EAI: *Enterprise Application Integration – Integração de Aplicações Empresariais.*

ebXML: *Electronic Business using eXtensible Markup Language – Negócios Eletrônicos utilizando Linguagem Extensível de Marcação.*

FTP: *File Transfer Protocol – Protocolo de Transferência de Arquivos.*

HTTP: *HyperText Transfer Protocol – Protocolo de Transferência de Hiper-Texto.*

J2EE: *Java 2 Platform, Enterprise Edition – Plataforma Empresarial Java 2.*

JAXP: *Java APIs for XML Processing – API Java para Processamento XML.*

JAXR: *Java API for XML Registries - API Java para Registros XML.*

JAX-RPC: *Java API for XML-Based RPC - API Java para RPC baseado em XML*

JCP: *Java Community Process – Processo da Comunidade Java.*

JNDI: *Java Naming and Directory Interface – Interface Java para Nomes e Diretórios.*

JSP: *JavaSever Pages – Tecnologia para Desenvolvimento de Aplicações Web.*

JSR-109: *Web Services for J2EE – Serviços Web para J2EE.*

RMI: *Remote Method Invocation – Invocação de Métodos Remotos.*

RPC: *Remote Procedure Call – Chamada Remota a Procedimentos.*

SAAJ: *SOAP with Attachments API for Java - API Java para SOAP com Anexos.*

SAX: *Simple API for XML – API Java para XML.*

SMTP: *Simple Mail Transfer Protocol – Protocolo de Transferência de e-mail.*

SOAP: *Simple Object Access Protocol - Protocolo baseado em XML para a troca de informações em um ambiente distribuído.*

TCP/IP: *Transmission Control Protocol / Internet Protocol – Protocolo de Transferência Confiável / Protocolo de Internet.*

UDDI: *Universal Description, Discovery and Integration – Descrição, Descoberta e Integração Universal.*

URI: *Uniform Resource Identifiers – Identificadores Uniformes de Recurso.*

W3C: *World Wide Web Consortium – Consorcio World Wide Web.*

WSDL: *Web Service Description Language – Linguagem de Descrição de Serviços Web.*

XSLT: *Extensible Stylesheet Language Transformation – Linguagem de Transformação de Documentos XML.*

XML: *Extensible Markup Language – Linguagem Extensível de Marcação.*

XSD: *XML Schema Definition – Definição de Esquemas XML.*

SUMÁRIO

1 INTRODUÇÃO

Com a necessidade cada vez maior das empresas integrarem dados e aplicações, tanto para possibilitar a troca de informações entre seus sistemas internos, como para com seus fornecedores, surge a necessidade de um padrão que torne este intercâmbio possível. Utilizando a Internet como meio de trocar estas informações foram criados os Serviços Web.

Serviços Web fornecem um mecanismo padrão de interoperabilidade entre aplicações, sendo possíveis suas execuções em diversas plataformas. São caracterizados por serem muito extensíveis, característica esta alcançada através do uso do protocolo XML (HENDRICKS,2002).

1.1 Motivação e Objetivos

Desde sua introdução com sucesso em 1999, os Serviços Web tornaram-se um padrão de desenvolvimento e distribuição de aplicações empresariais portáteis e distribuídas. Um importante fator para o sucesso dos Serviços Web, foi a plataforma J2EE, desenvolvida através de um processo aberto, o Processo da Comunidade Java o JCP *(Java Community Process)*. Este processo abrangeu um grande número de fornecedores de computação, desejando que a plataforma chegasse o mais próximo das exigências empresariais (SINGH, 2004).

Utilizando-se da plataforma J2EE, que dá suporte ao desenvolvimento de Serviços Web através da linguagem de programação Java, executando sobre o sistema operacional linux, será realizado o desenvolvimento de um Serviço Web. Onde será programado apenas o lado do servidor.

Este Serviço Web oferecerá métodos que possibilitem a recuperação e centralização das informações sobre prontuários de pacientes de Instituições de Saúde.

1.2 Organização da Monografia

Este trabalho visa explorar os padrões envolvidos na tecnologia de Serviços Web descritos no capítulo *1 – Infra-estrutura de Serviços* Web, a plataforma J2EE e recurso disponíveis no auxilio ao desenvolvimento de Serviços Web mostrados no capítulo *2 - Plataforma J2EE e Serviços Web*, a implementação de uma aplicação na qual haja a necessidade de troca de informações de interesses comuns entre organizações, independentes de suas plataformas de hardware ou software descrito no capítulo 3 - *Implementação do Serviço Web*. A conclusão abordando a implementação de um Serviço Web utilizando-se da plataforma J2EE de desenvolvimento, analisando as características do desenvolvimento nesta plataforma.

2 INFRA-ESTRUTURA DE SERVIÇOS WEB

Os Serviços Web são compostos por uma infra-estrutura de tecnologias que tornam seu desenvolvimento possível.

Entre estas tecnologias encontram-se: Linguagem XML, Protocolo SOAP, Especificação WSDL e Padrão UDDI; que trabalham em conjunto para assegurar a interoperabilidade, independência de plataforma e extensibilidade dos Serviços Web.

Neste capítulo serão abordadas estas tecnologias, suas principais características e como funcionam.

2.1 O que são Serviços Web?

Segundo o W3C (2005) organização que estabelece os padrões para Serviços Web:

> "O Serviço Web é um software identificado por um URI nos quais interfaces e ligações são definidas e descritas utilizando XML. Estas definições podem ser descobertas por outros softwares. Estes softwares podem então interagir com o Serviço Web do modo prescrito por suas definições, usando mensagens baseados em XML através de protocolos da Internet."

Serviços Web são softwares que possibilitam a intercomunicação de outros softwares através de suas interfaces e padrões já estabelecidos. Uma característica importante é o fato de independência de plataforma, pois utiliza padrões abertos já estabelecidos em toda sua infra-estrutura.

Na arquitetura de Serviços Web, existem algumas tecnologias principais que tornaram seu desenvolvimento possível, são elas, XML uma linguagem de marcação de dados, UDDI *(Description, Discovery and Integration)* especificação responsável pela localização de objetos, WSDL *(Web Service Description Language)* responsável pela descrição de serviços e SOAP *(Simple Object Access Protocol)* protocolo básico para troca de mensagem (HENDRICKS, 2002).

Serviços Web possuem um modelo conceitual típico, no qual é possível descrever todas as entidades e seus papéis, e operações por eles executadas..

Na figura 1 tem-se estas principais entidades trabalhamdo em conjunto, cada uma possuindo um papel no modelo dos Serviços Web. Existem três papéis principais, como mostrados a seguir:

Provedor de serviços - É responsável pela criação dos serviços que serão consumidos pelos clientes. Utiliza o protocolo WSDL para fornecer a descrição e a localização do serviço.

Registro do Serviço – Local onde o provedor de serviços publica seus serviços para posterior pesquisa e localização pelo cliente. Utiliza o protocolo UDDI responsável pela especificação de localização de objetos.

Consumidor do Serviço – Uma aplicação que necessite utilizar de algum serviço disponibilizado pelo provedor de serviços. Utiliza o protocolo SOAP para se comunicar com provedor e registro do serviço.

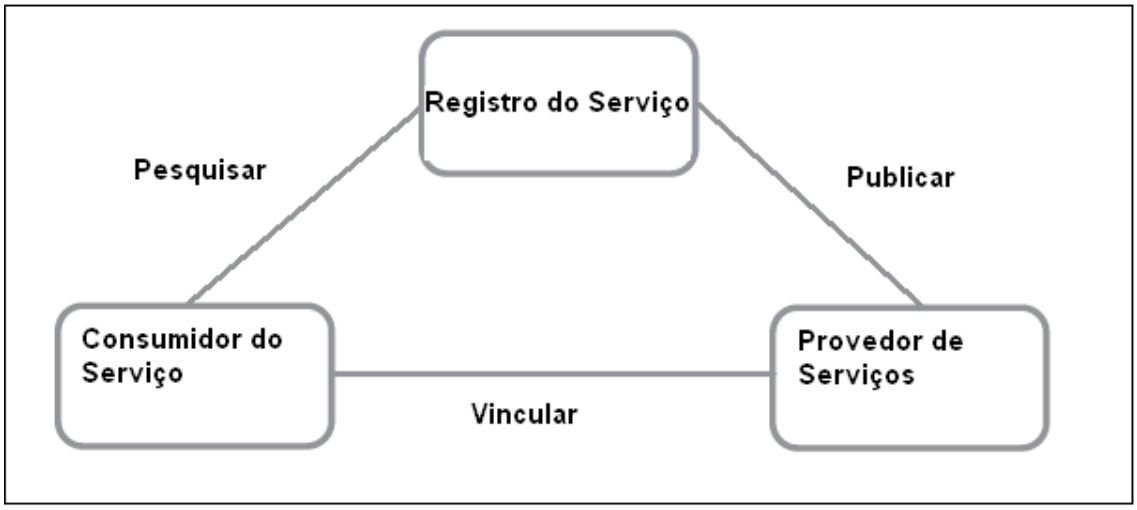


Figura 1 – Modelo conceitual de Serviços Web (HENDRICKS, 2002)

2.2 Linguagem XML

XML é essencialmente uma linguagem de marcação de dados possibilitando que estruturas hierárquicas de dados representadas através de programação, possam ser representadas através de um documento de texto com marcações. Por ser uma linguagem de marcação de dados, XML utiliza identificadores para demarcar partes de dados. Cada

identificador tenta passar um sentido para o dado associado a ele, esta é uma forma de transformar dados em informações (SINGH, 2004).

A especificação XML foi desenvolvida para ser legível pelos humanos, algo que contribuiu para sua aceitação por parte dos usuários. XML é suportado para comunicação tanto entre humanos quanto por computadores, possuindo assim uma transparência e independência de plataforma, características não observadas em sistemas baseados em comunicação binária.

XML possui uma gramática, ou seja, uma forma de representar a estrutura esperada em um documento XML. A Definição de Tipos de Documento (DTD), uma especificação do núcleo da especificação XML, possibilita estas definições de tipos relacionadas a um domínio. Esta característica é algo muito importante para padronização da estrutura de informações trocadas entre empresas, em suas operações de negócios.

Em resumo, XML pode ser considerada uma metalinguagem usada para a definição de outras linguagens de marcação. Os seus identificadores não são suficientes para representar documentos completamente altos descritos. Através de esquemas escritos como DTDs, ou utilizando outra linguagem de definição de esquemas como a W3C *(XML Schema Definition)* XSD, sua completa descrição torna-se possível (SINGH, 2004).

No Quadro 1 tem-se um exemplo da estrutura da linguagem XML onde <nome> representa um elemento ”nome”, e tipo=”celular” representa um atributo do elemento telefone.

```
<contato codigo=”11”>
    <nome>Paulo da Silva</nome>
    <emai>paulo@email.com.br</email>
    <telefone tipo=”celular”>
     <area>11</area>
     <numero>999 9999-9999</numero>
   </telefone>
</contato>
```

Quadro 1 - Exemplo de um documento XML

2.3 Linguagem de definição de Esquemas (XSD)

Publicada em 2001 como uma recomendação W3C, essa linguagem expressa uma forma de compartilhar vocabulários permitindo aos computadores seguirem regras criadas por pessoas. Um meio de definir estruturas, conteúdo e semântica de um documento XML (SPERBERG-MCQUEEN,2005).

O XSD fornece uma linguagem de esquema mais flexível que a DTD, possibilitando a representação de sintaxe de documentos XML em linguagem XML e a manipulação de sintaxe para tipos de dados complexos.

Tendo um documento XSD definido, diz-se que um arquivo XML é válido para este esquema se condiz com as regras do esquema, sendo chamado de instância do esquema.

O XSD suporta tipos simples, que não possuem sub-elementos, sejam outros elementos ou atributos, e tipos complexos que possuem sub-elementos.

Para a utilização de tipos é necessária sua definição e posterior declaração. O processo de definição cria novos tipos, sejam eles simples ou complexos. A declaração possibilita que elementos e atributos com nomes e tipos específicos sejam utilizados nas instancias do documento.

Existem na linguagem XSD tipos pré-definidos, entre eles, `String, token, byte, integer, long, short`.

O XSD possibilita o controle da ocorrência de elementos no arquivo XML, podendo ser definidas a máxima e a mínima ocorrência de certo elemento (SPERBERG-MCQUEEN,2005).

No Quadro 2, a estrutura de um documento DTD e XSD de exemplo:

```
//DTD
<!ELEMENT contato (nome, email, telefone)>
<!ATTLIST contato codigo NMTOKEN #REQUIRED>
.................................................
//XSD
<xsd:schema    xmlns:xsd=".../XMLSchema">
<xsd:element name="contato">
    <xsd:complexType>
        <xsd:attribute name="codigo"  use="required">
```

Quadro 2 - Estrutura exemplo de um documento DTD e XSD

2.4 Protocolo SOAP

O SOAP é um protocolo de computação superficial distribuído que permite a troca de informações em um ambiente descentralizado e distribuído (HENDRICKS, 2002).

O SOAP é o protocolo padrão para a troca de mensagens em Serviços Web J2EE e muito utilizado para este fim em Serviços Web de modo geral. SOAP possibilita um meio de comunicação entre aplicações, muito utilizado em sistemas para comunicação entre empresas *Business-to-Business* (B2B), e Integração de Aplicações Empresariais (EAI), ambos focados em integração de aplicações e compartilhamento de dados (WEERAWARANA,2005).

Utiliza a linguagem XML baseada em texto, algo que lhe dá uma grande vantagem em relação aos protocolos existentes que utilizam arquivos binários, tais como, (CORBA), (RMI), (DCOM). Esta característica o torna altamente interoperável por meio de várias plataformas de hardware, redes, sistemas operacionais e linguagens de programação. O fato de o SOAP poder ser transportado através do protocolo de transporte HTTP, permite a utilização de infra-estruturas já existentes, como servidores *proxy* e *firewalls* (HENDRICKS, 2002).

O SOAP é considerado um protocolo superficial por não implementar várias funções presentes em outros protocolos para aplicações distribuídas, tornando-o menos complexo.Pode ser visto como uma linguagem de marcação XML com regras específicas, para um propósito básico que é a troca de dados entre redes.

A especificação SOAP é composta por duas partes, o encapsulamento de mensagens e chamadas a procedimentos remotos (RPC).

O **encapsulamento de mensagens** define como embutir chamadas e respostas a procedimentos dentro da mensagens, para tornar possível sua chamada em sistemas remotos.

A **chamada a procedimentos remotos** permite que o SOAP forneça uma abstração na camada do protocolo, para protocolos de computação distribuídos dentro de uma Intranet ou na Internet. Essa capacidade de abstração fornece sistemas remotos com acesso a objetos através de plataformas, sistemas operacionais e ambientes que seriam de outra maneira incompatíveis (HENDRICKS, 2002).

Uma instância de um documento XML SOAP, é chamado de mensagem SOAP e é transportada como carga de algum outro protocolo de rede. O protocolo mais utilizado para troca de mensagens é o HTTP, utilizado por navegadores para acessarem conteúdos HTML. A grande diferença das mensagens SOAP é que não podem ser vistas com os navegadores como ocorre com o HTML, ela é utilizada para comunicação entre aplicações utilizando-se do protocolo HTTP como um dos meios para realizar esta troca de mensagens (WEERAWARANA,2005).

Mensagens SOAP podem também ser transportadas por e-mail utilizando o protocolo SMTP, ou por outros protocolos de redes como (FTP) e (TCP/IP). Entretanto a WS-I, órgão responsável pela interoperabilidade de Serviço Web, define o HTTP como protocolo padrão para troca de mensagens SOAP.

Na Figura 2 ilustra-se como o SOAP pode ser transportado por vários protocolos entre aplicações em uma rede. Uma Aplicação C comunica-se com uma Aplicação B utilizando o protocolo (SMTP), a Aplicação B comunica-se com a Aplicação A utilizando o protocolo HTTP e a Aplicação C pode comunica-se com a Aplicação A utilizando o protocolo (TCP/IP).

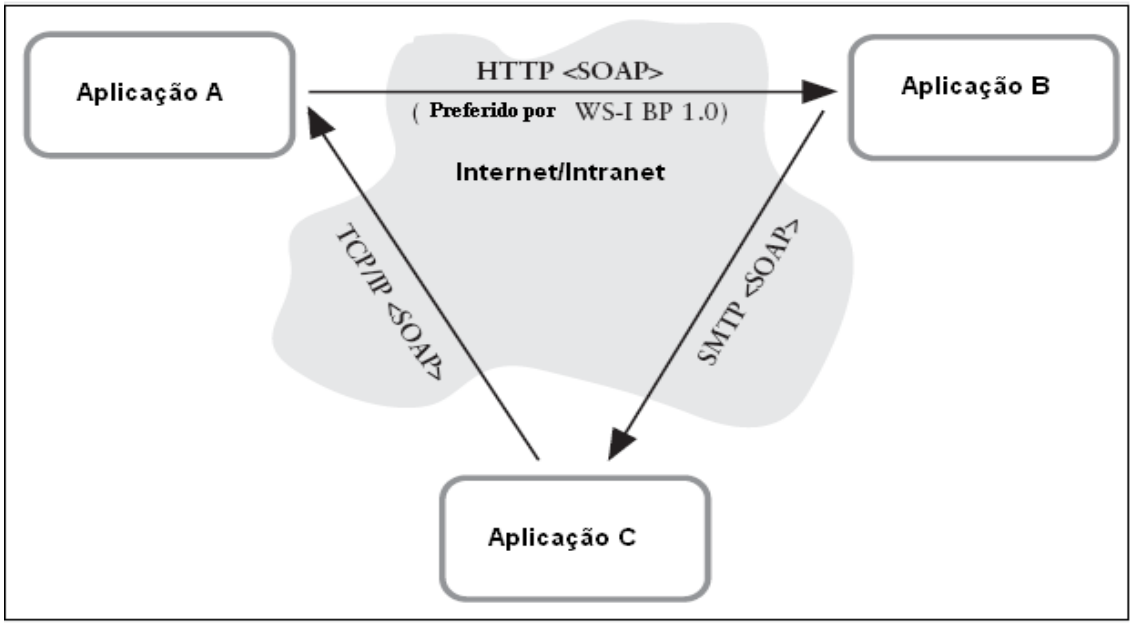


Figura 2 – SOAP sobre TCP/IP, HTTP e SMTP (WEERAWARANA,2005)

2.4.1 Estruturação de protocolo SOAP

O protocolo SOAP utiliza a linguagem XML para codificar todas as suas mensagens.

As mensagens SOAP devem conter os *namespaces* característicos da XML em todos seus elementos e atributos específicos definidos pelo SOAP. A utilização de *namespaces* no SOAP é importante para que elementos que forem definidos pelo usuário não entrem em conflito com os elementos do SOAP padrão.

Todos os atributos do SOAP padrão recebem um prefixo com o identificador de *namespace* SOAP-ENV, que está associado ao *namespace* `http://schemas.xmlsoap.org/soap/envelope.`

Uma mensagem SOAP é um documento XML contendo até três partes: o envelope SOAP, uma parte obrigatória do protocolo, um cabeçalho opcional e o corpo do documento que também é obrigatório.

NA Figura 3 descreve-se graficamente a estrutura de uma mensagem SOAP.

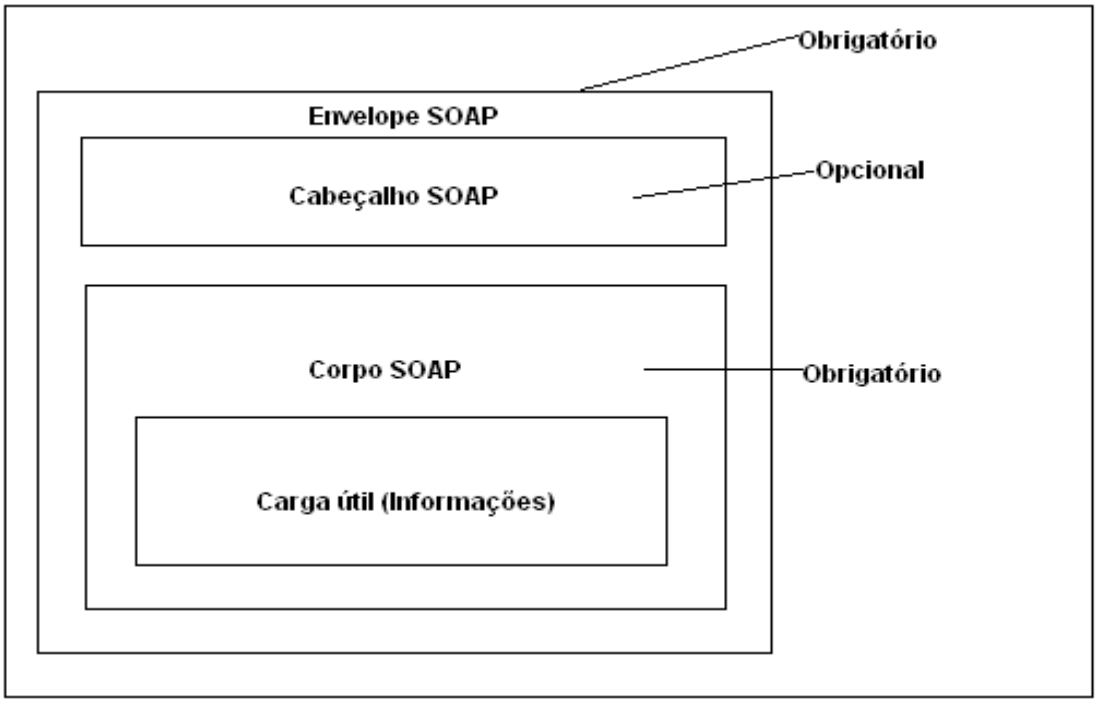


Figura 3 – Composição de uma mensagem SOAP (HENDRICKS, 2002)

O envelope SOAP é o elemento raiz de uma mensagem SOAP. Toda a estrutura da mensagem SOAP está dentro dele. Dentro do envelope possui um cabeçalho opcional e um corpo obrigatório. Para representar um envelope em uma mensagem SOAP é utilizado o elemento `<Envelope>.`

No Quadro 3 apresenta-se um exemplo de envelope SOAP:

```
<SOAP-ENV:Envelope
      <SOAP-ENV:Header>
            ... //Esta parte é opcional.
      </SOAP-ENV:Header>
      <SOAP-ENV:Bodyr>
            ... //Esta parte é obrigatória.
      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Quadro 3 - Estrutura de um envelope SOAP

O cabeçalho SOAP é um mecanismo que facilita a inserção de recursos na mensagem SOAP. Recursos como autenticação e gerenciamento de transações em uma mensagem SOAP. Através do elemento `SOAP-ENV:Header` um cabeçalho SOAP é especificado.

O cabeçalho SOAP pode conter elementos filhos, representados como entradas de cabeçalho dentro da especificação SOAP.

O Quadro 4 descreve como um cabeçalho SOAP pode ser specificado para realizar uma autenticação.

```
<SOAP-ENV:Envelope
      xmlns:SOAP-ENV=”http://schemas.xmlsoap.org/soap/envelope/”
      SOAP-
ENV:encodingStyle=”http://schemas.xmlsoap.org/soap/encoding/”>
      <SOAP-ENV:Header>
          <a:authentication                xmlns                :
a=”http://www.site.com/soap/authentication”>
                    <a:username>usuario</a:username>
            <a:password>senha</a:password>
          </a:authentication>
      </SOAP-ENV:Header>
      <SOAP-ENV:Body>
            ... //Esta parte é obrigatória.
      </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Quadro 4 - Exemplo de utilização do cabeçalho SOAP para autenticação

O corpo de uma mensagem SOAP contém todas as informações, ou seja, a carga útil que será recebida pelo receptor da mensagem. Através do elemento `<Body>` é especificado o

corpo de uma mensagem SOAP. Caso um elemento de cabeçalho esteja presente na mensagem, deverá ser inserido um elemento `<Body>` logo após o elemento `<Header>`.

O SOAP fornece recursos para tratamento de erros em aplicações, através do elemento `<Fault>`. É possível também utilizar tipos de dados da especificação *"XML Schema Part2: Datatypes"*, uma especificação da W3C (HENDRICKS,2002).

Através de vinculação é possível utilizar o protocolo SOAP sobre diversos protocolos de transporte, como HTTP e SMTP (WEERAWARANA,2005).

2.5 Especificação WSDL.

A especificação WSDL (CHRISTENSEN, 2001), define um meio de descrever Serviços Web. Utilizando a linguagem XML a especificação possui um esquema que descreve a estrutura da linguagem XML que cada documento deve conter.

Uma importante característica da tecnologia WSDL, é permitir que as aplicações se comuniquem entre si sobre qualquer rede de comunicação, sem a obrigatoriedade de compartilhar a mesma linguagem de programação (HENDRICKS, 2002).

Segundo CHRISTENSEN (2001):

> "WSDL é um formato XML para descrever serviços de rede como um conjunto de extremidades operando sobre mensagens, contendo informações orientadas a documento ou a procedimento. As operações e as mensagens são descritas resumidamente, e então transportadas para um protocolo de rede concreto e um formato de mensagem para definir uma extremidade. Extremidades relacionadas concretamente são combinadas em extremidades resumidas (serviços). WSDL é extensível permitindo descrição de extremidades e suas mensagens, independente do formato de rede ou protocolo usado para a comunicação."

A especificação WSDL foi definida em conjunto com a IBM, Microsoft e Ariba, sendo posteriormente submetida a W3C para ser padronizada.

O documento WSDL é responsável por apresentar as interfaces externas e a localização de serviços, sendo assim possível oferecer os serviços para qualquer cliente.

2.5.1 Componentes do Serviço Web definidos pela WSDL

Com o objetivo de oferecer maior flexibilidade, a WSDL define os vários componentes de um serviço, que podem ser então reutilizados para definir diversos serviços. Entre os componentes que compõem o serviço estão os descritos a seguir:

- Tipos de dados como `string, int, object` entre outros.
- Os parâmetros de entrada e saída de um serviço. Um parâmetro de chamada de um serviço é denominado mensagem.
- O relacionamento entre parâmetros de entrada e saída. A assinatura do método, também conhecida como operações.
- Agrupamento lógico de operações. Um determinado objeto que é um agrupamento de métodos é conhecido como um tipo de porta.
- O protocolo a ser usado para acessar os métodos de um objeto. As opções disponíveis na especificação WSDL atualmente são SOAP, HTTP e MIME. Este processo é conhecido como um vínculo.
- O endereço do Serviço Web. Descreve a localização do serviço na rede.

2.5.2 Elementos da especificação WSDL

O elemento `<types>` define quais serão os tipos de dados aceitos pelo Serviço Web, estes dados são utilizados com freqüência em definições XML. Se não existir a definição XML, indica que o serviço está utilizando apenas os tipos de dados básicos, como *strings inteiro*s e outros.

O elemento `<message>` é uma unidade conceitual de dados que podem ser trocados. Uma mensagem pode ser composta de alguns argumentos, no caso da chamada de um método todos os parâmetros do método são representados por uma única mensagem. Para isso, cada elemento `<message>` contém um ou mais elementos `<part>` que formam as partes reais da

mensagem. Cada elemento <part> se refere a um tipo definido no elemento <types> do documento (HENDRICKS, 2002).

Após a mensagem ter sido definida, através do elemento <message> é necessário especificá-la em uma seqüência útil. A especificação WSDL define esta seqüência como uma operação.

Existem quatro tipos possíveis de operações: unidirecional, solicitação-resposta, pedido-resposta e notificação.

A operação unidirecional define uma mensagem que é enviada de um cliente para um serviço, sem que ocorra nenhuma resposta desse serviço.

A solicitação-resposta é o tipo mais comum, nela um cliente envia uma solicitação a um serviço e recebe uma mensagem de resposta, como resultado desta solicitação.

A operação pedido-resposta define uma mensagem enviada do serviço para o cliente, resultando em uma mensagem que é enviada do cliente para o serviço.

Na operação notificação a mensagem é enviada do serviço para o cliente apenas.

O tipo de operação mais utilizado atualmente é o tipo solicitação-resposta, que pode ser visto como uma chamada remota a procedimentos RPC. A RPC é uma chamada de funções pelos limites de processos que utilizam alguns parâmetros e retornam um resultado.

Na operação solicitação-resposta, a solicitação por um serviço resulta no retorno de uma resposta. Essa operação é composta por uma mensagem de entrada e outra de saída. As mensagens utilizadas nesta operação devem ser definidas no elemento <message> no documento WSDL. Uma ou mais operações podem ser agrupadas em um elemento <portType>.

O elemento <binding> descreve o mecanismo usado por um serviço para se comunicar com um cliente. É o elemento mais complexo, por permitir que mais elementos possam se inclusos nele.

A especificação WSDL foi descrita para ser independente do mecanismo utilizado para acessar um serviço. O protocolo SOAP é o mecanismo mais utilizado para garantir este acesso,

porém a especificação ainda inclui dois outros mecanismos: o HTTP e o MIME (CHRISTENSEN,2005).

O HTTP destina-se principalmente a chamar um Serviço Web a partir de um navegador Web, por meio de procedimentos, tais como, `GET e POST`.

O vínculo MIME pode ser usado junto com o SOAP ou o vínculo de HTTP para adicionar recursos de codificação MIME a um serviço.

Em todas as informações de vínculo se repetem as operações e suas respectivas mensagens de entrada e saída “tipo de porta”, e são adicionadas informações relativas ao protocolo utilizado em particular.

Na Figura 4 descrevem-se os principais elementos WSDL que podem ocorrer em um documento.

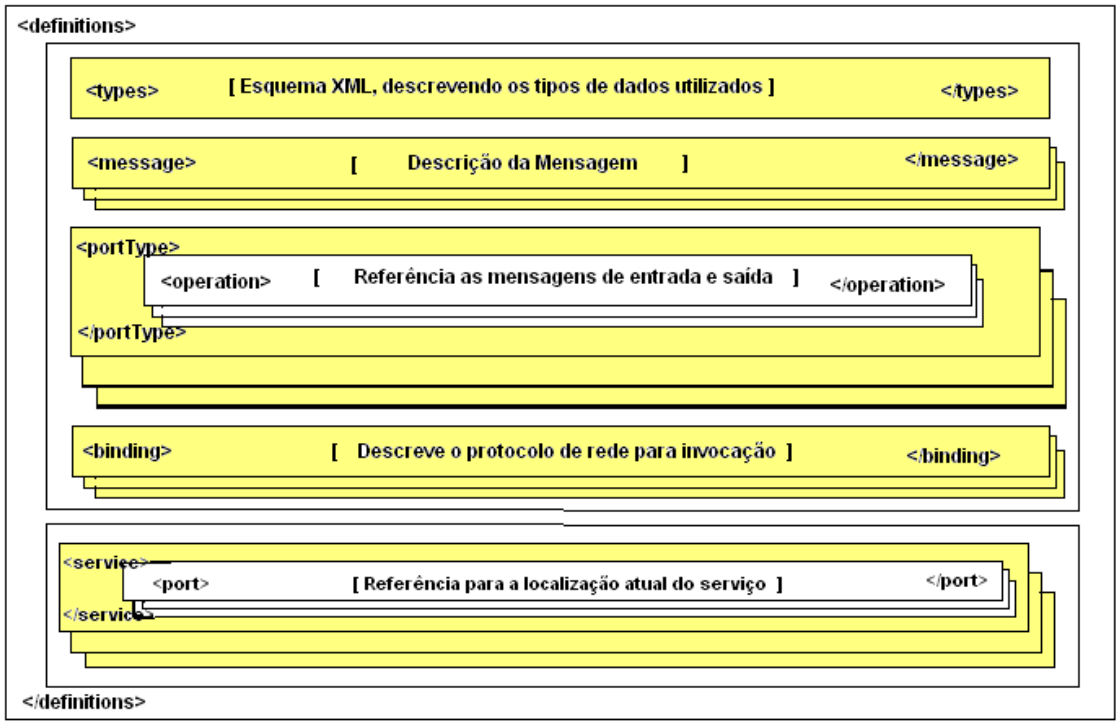


Figura 4 – Principais elementos de um documento WSDL (HENDRICKS, 2002)

2.6 Padrão UDDI

Um fator importante para os Serviços Web é a existência de um mecanismo para registro e descoberta de serviços.

O serviço deve se comunicar com os clientes, independente da plataforma onde está sendo utilizado e da linguagem de programação, baseando-se no conceito de Arquitetura Orientada a Serviço SOA, a qual define três papéis, o requisitante do serviço, o provedor do serviço e o *broker* do serviço.

O requisitante do serviço é uma entidade interessada em algum serviço fornecido pelo provedor de serviço

O *broker* tem o papel de ligação entre as entidades provedor de serviço e requisitante do serviço. Ele geralmente contém as referências dos serviços disponíveis.

Os Serviços Web são geralmente armazenados em registros UDDI e contém especificações que descrevem como um registro armazena dados e como pode ser acessado. O padrão UDDI foi criado, em um consórcio entre Ariba, IBM e Microsoft, porém para maior aceitação, foi criada uma comunidade que discute a especificação e cria padrões a partir dela, o consórcio OASIS-UDDI padrões.

UDDI é um dos padrões mais importantes para o sucesso dos Serviços Web, pois cria um padrão independente de plataforma, permitindo que companhias e aplicações rapidamente, facilmente e dinamicamente encontrem e usem Serviços Web através da Internet (OASIS, 2005).

2.6.1 Especificações UDDI.

O padrão UDDI possui quatro especificações principais, a de estrutura de dados, a API *(Application Program Interface)* do programador, a replicação e o operador.

A especificação de estrutura de dados descreve quais são os tipos de dados armazenados na UDDI, é baseada na linguagem XML permitindo independência de plataforma e linguagem

de programação. Essa estrutura pode ser descrita através de um esquema XML, disponibilizado pela OASIS, órgão responsável pela sua padronização.

Na especificação da API do programador possui informações sobre como acessar um registro UDDI. Existem dois tipos, as funções de publicação que são utilizadas para criar e atualizar entradas existentes no registro, e as funções de pesquisa que são de leitura, permitindo que entradas sejam consultadas via programação.

A especificação de replicação descreve como os registros trocam informações entre si. Essa característica é importante para que interessados em desenvolver seus registros individuais, possam integrá-los com registros já existentes.

A especificação do operador define políticas de segurança e de gerenciamento de dados, para quem estiver desenvolvendo ou executando um registro UDDI. Porém não exige que se siga uma política determinada; solicita que cada operador publique as políticas permitidas e as impostas.

2.6.2 Estrutura de dados UDDI.

O registro UDDI armazena informações sobre negócios e os serviços oferecidos. Não possui informações sobre Serviços Web especificamente, mas foi projetado para manter informações diversas sobre um negócio. Fornece uma fonte de informações sobre serviços e negócios.

A composição de sua estrutura é semelhante a um catálogo telefônico, podendo ser aplicada à publicação e recuperação de dados UDDI. A estrutura baseia-se na seguinte forma:

- As Páginas Brancas fornecem informações sobre nomes, telefones, endereço e o negócio.
- As Páginas Amarelas fornecem listagens comerciais de acordo com o negócio, e também entradas classificadas por um tipo de negócio ou a indústria que opera.

- As Páginas Verdes contém serviços oferecidos por cada negócio, contendo todas as informações técnicas envolvidas na interação com o serviço. Informações como a utilização do serviço, seus parâmetros, valores de extremidades.

2.6.3 Elementos UDDI

O elemento `<tModel>` é um dos fundamentais, representa uma especificação técnica em um registro UDDI. Descreve todo o tipo de informação, podendo ser padrões para troca de dados entre negócios ou a definição de uma interface de um Serviço Web.

Se for criado um Serviço Web e sua descrição em WSDL, esta definição será armazenada com um elemento `<tModel>.`

O elemento raiz da hierarquia de dados UDDI `<businessEntity>`, contém informações sobre negócios, tais como, nome, endereço e telefone. Permitindo ser categorizada para futura localização.

Na Figura 5 descrevem-se os principais elementos contidos no UDDI.

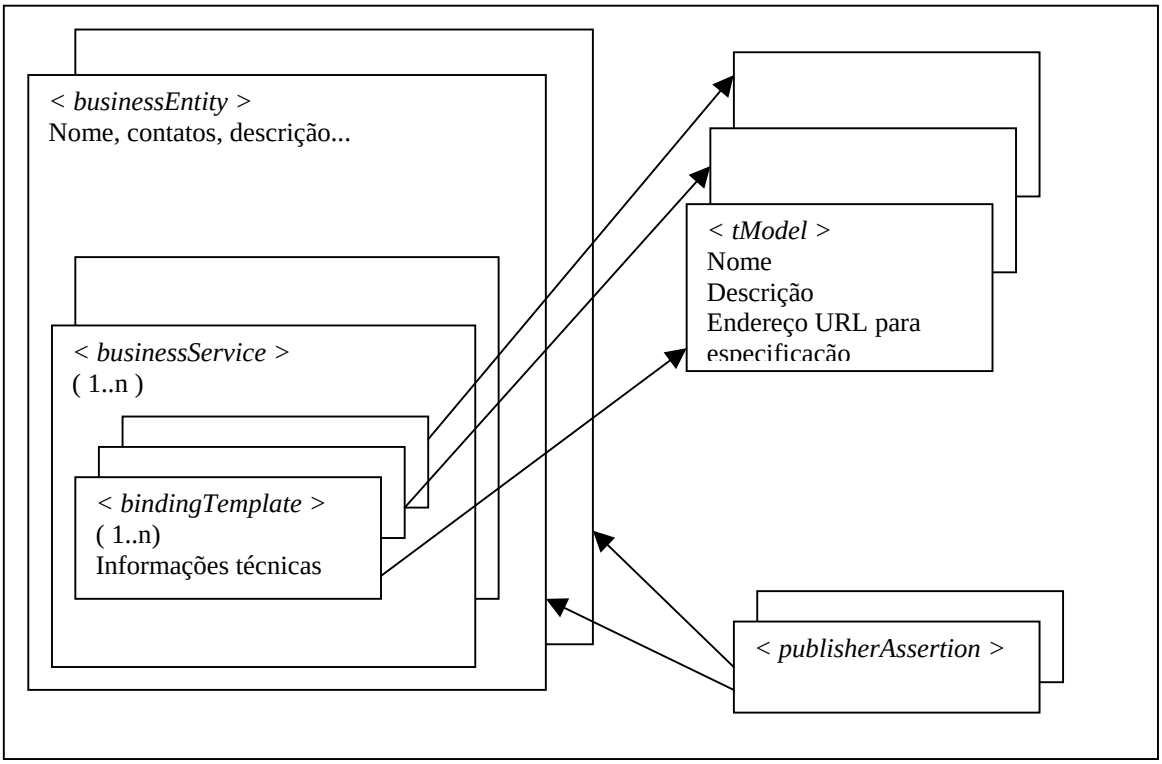


Figura 5 – Principais elementos UDDI (HENDRICKS, 2002)

3 PLATAFORMA J2EE E SERVIÇOS WEB

A plataforma de desenvolvimento J2EE foi introduzida no ano de 1999, e tornou-se um padrão integrado para o desenvolvimento de aplicações empresariais, portáteis e distribuídas.

Um dos fatores que contribuiu para a grande aceitação da plataforma J2EE foi ter sido desenvolvida através de um processo aberto, o Processo da Comunidade Java JCP *(Java Community Process)*.

Este processo aberto contou com o apoio de diversos fornecedores de tecnologia, unidos no objetivo de conseguir abranger através da plataforma o maior número de requisitos empresarias possíveis.

Com a grande popularização dos Serviços Web, a plataforma J2EE adicionou componentes e tecnologias específicas para o desenvolvimento de Serviços Web.

Estes novos componentes J2EE e tecnologias satisfizeram as necessidades da maioria da empresas. Permitiram às empresas exporem suas aplicações existentes como Serviços Web, e também a desenvolver novos serviços (SINGH, 2004).

3.1 A plataforma J2EE 1.4

A plataforma J2EE foi projetada para dar suporte ao desenvolvimento de Serviços Web tanto no lado do cliente quanto no lado do fornecedor do Serviço Web.

Foi projetada para facilitar o desenvolvimento de aplicações multicamadas e Serviços Web. A plataforma inclui uma camada do cliente, uma ou mais camadas intermediarias e uma camada Servidor. Também define um meio padronizado de comunicação entre as camadas, permitindo que partes da aplicação estejam em diferentes camadas (SINGH, 2004).

O modelo de desenvolvimento multicamadas permite que várias partes de uma aplicação rodem em sistemas diferentes. Também permite que a plataforma suporte uma grande variedade de clientes.

Os tipos de clientes suportados pela plataforma são: navegadores web, clientes sem fio (Wireless), interfaces gráficas com usuário Java GUI *(Ggraphical User Interface)*, diversos clientes não desenvolvidos em Java.

A plataforma J2EE também suporta o desenvolvimento baseado em componentes de aplicações e serviços. As partes da lógica da aplicação que reside em diferentes camadas, podem fazer uso de diferentes componentes fornecidos pela plataforma para diferentes camadas. Deste modo o desenvolvimento baseado em componentes favorece a reusabilidade, sendo assim componentes são módulos discretos que podem ser usados virtualmente por qualquer outro componente quando necessário (SINGH, 2004).

Para alcançar um caminho padrão de interatividade entre componentes, a plataforma introduziu um conceito de gerenciamento de *containers* baseado em componentes. Componentes rodam dentro de *containers*, que são ambientes de tempo de execução padronizados que fornecem serviços específicos para os componentes, deste modo, alcança portabilidade da aplicação através da plataforma de implementação (SINGH, 2004).

O padrão J2EE é definido através de um conjunto de especificações existentes, entre elas, *Enterprise Java Beans EJB, Java Servlet, JavaSever Pages JSP* , API Java para processo XML baseado em RPC JAXP. Essas especificações definem e certificam uma arquitetura de plataforma que possibilita o desenvolvimento de aplicações e Serviços Web, distribuídos, portáveis e interoperáveis.

A plataforma J2EE também suporta a especificação da organização responsável pela Inter-operabilidade de Serviços Web a “WS-I Basic Profile” , que abrange especificação para isso por meio de plataformas de desenvolvimento, sistemas operacionais e linguagens de programação.

Segundo SINGH (2004), a plataforma J2EE 1.4 inclui especificações e tecnologias que suportam o protocolo SOAP, a linguagem WSDL e a especificação UDDI. Incluindo tecnologias que possibilitam uso padronizado de especificações para Serviços Web dentro da plataforma Java: API Java para RPC Baseado em XML (JAX-RPC), Serviços Web para J2EE

(JSR-109), SOAP com Anexos API para Java (SAAJ), API Java para registro de XML (JAXR), e API Java para Processamento XML (JAXP).

Na Figura 6 mostra-se o inter-relacionamento das tecnologias envolvidas na plataforma J2EE 1.4, tais como, Containers de Aplicações que se comunicam com *Containers Web e EJB*, descrevendo suas principais APIs..

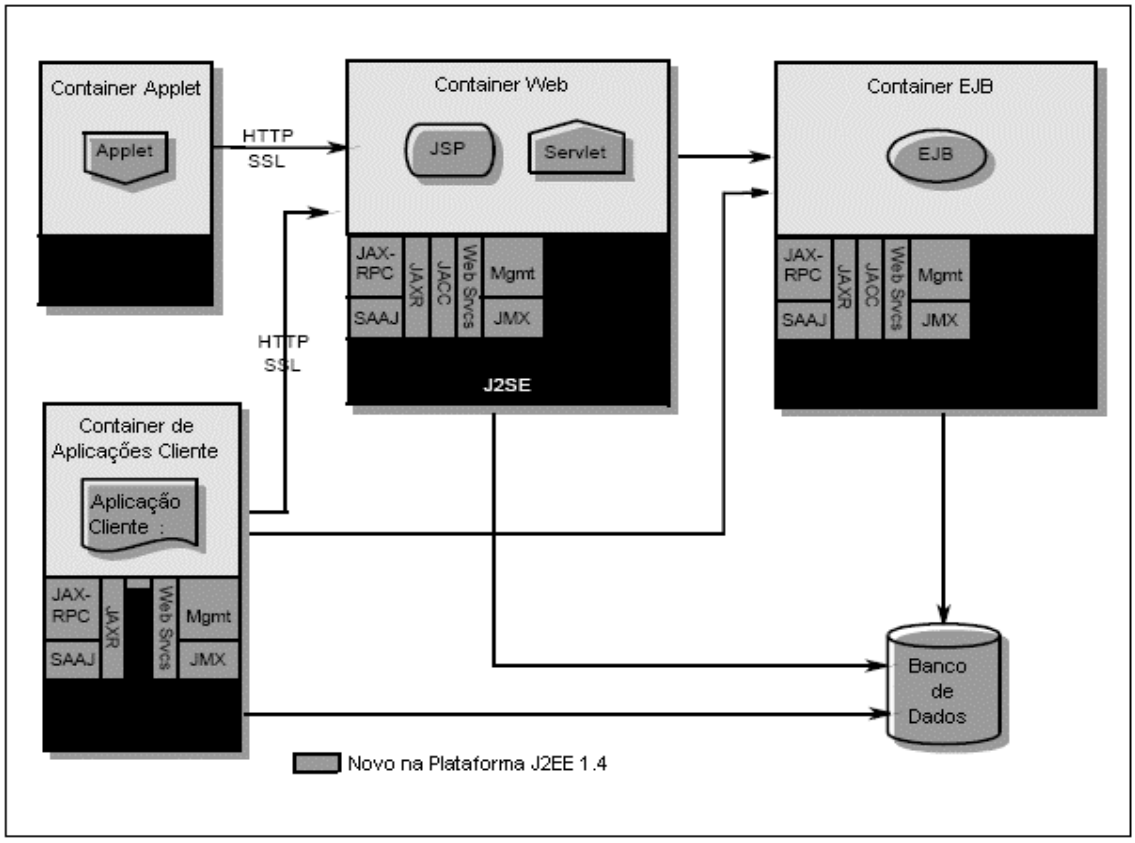


Figura 6 – Arquitetura da plataforma J2EE 1.4 (SINGH, 2004)

3.2 Tecnologias da plataforma J2EE 1.4

A plataforma J2EE consiste de tecnologias que suportam o desenvolvimento de aplicações e serviços empresarias distribuídos.

A plataforma é composta por três conjuntos de tecnologias básicas os componentes, os serviços e a comunicação.

3.2.1 Tecnologia de Componentes

Os componentes são utilizados pelos desenvolvedores para criar as partes essências de aplicações empresariais e Serviços Web, como por exemplo, a regra de negócio do serviço.

A utilização da tecnologia baseada em componentes para o desenvolvimento das partes essenciais do serviço, promove um desenvolvimento modular e favorece o reuso de módulos.

As diversas tecnologias de componentes presentes na plataforma J2EE são as seguintes:

- Componente Cliente – A plataforma fornece suporte para diferentes tipos de clientes interagirem com componentes no lado do servidor. Por exemplo, os Clientes Applets são baseados em Java e normalmente rodam dentro de um navegador Web e possuem total acesso aos recursos da plataforma J2SE.

Aplicações clientes são executadas em seus próprios *containers* e possuem total acesso a serviços J2EE, como buscas JNDI *(Java Naming and Directory Interface)*, e toca de mensagens assíncronas, entre outros. Estes clientes podem interagir com os componentes Web e EJB no lado do servidor de aplicações.

- Componente Web – Os componentes Web fornecem resposta para requisições recebidas via HTTP. A plataforma J2EE define dois tipos distintos de componentes Web, são eles, componentes Servlets e JSP.

Os componentes Servlets, ampliam as funcionalidades de um Serviço Web de maneira eficiente. Com Servlets o desenvolvedor pode mapear um conjunto de URLs para um conjunto de Servlets. Como resultado para este mapeamento uma requisição HTTP para uma destas URLs invoca o Servlet mapeado, que processa a requisição e envia uma resposta. Os Servlets também podem ser expostos como Serviços Web.

Os componentes JSP possibilitam a geração de conteúdo dinâmico.

- EJB *(Enterprise Java Beans Component)* componentes são desenvolvidos com lógica de negócio em mente, são componentes escalábeis, transacionais e seguros.

Os componentes *Session bean*, um tipo de EJB componente, usualmente fornecem um serviço para apenas um único cliente e seu status não pode ser recuperado após a ocorrência de um erro.

Os componentes *Entity Bean* são a representação em forma de objetos de dados contidos em fontes de dados. Este componentes gerenciam persistência de dados.

Os componentes *Message-drive bean*, permitem aos clientes acessarem lógica de negócio contidas dentro de componentes empresarias de maneira assíncrona.

O desenvolvimento de aplicações e Serviços Web utilizando estas tecnologias assegura uma padronização de aplicações ou serviços. Esta padronização favorece a portabilidade e reuso de aplicações e serviços.

3.2.2 Plataforma e serviços de *container*.

A tecnologia de componentes da plataforma J2EE depende de *containers* para funcionar corretamente. Para assegurar que componentes são portáveis, a plataforma J2EE solicita uma confirmação a um provedor da plataforma, para tornar alguns serviços disponíveis. Entre os serviços requeridos pela plataforma tem-se:

- Serviços de Nome – Permitem um acesso a componentes dentro de um ambiente de nomes. Estes componentes podem ser customizados sem necessidade de alterar código fonte.
- Serviço de Montagem – Este serviço permite a alteração de comportamento de componentes no momento da montagem sem a necessidade de alterar o código fonte.
- Serviço de Transação - Um serviço de transação elimina do desenvolvedor de componentes a necessidade de construir código para manipular funções transacionais tais como acesso a múltiplos usuários e recuperação de falhas.
- Serviços de Segurança – Serviço de segurança assegura que componentes e fontes sejam acessadas somente por autorizados. O serviço de segurança

também fornece autenticação e confidencialidade, entre outros, para os usuários.

3.2.3 Comunicação.

Segundo SINGH (2004):

> "A plataforma J2EE, além de componentes e serviços, também requer tecnologias de comunicação padrão. Estas tecnologias unem componentes e serviços da plataforma, tornando J2EE uma plataforma de desenvolvimento integrado e um padrão para desenvolvimento aplicações empresariais e Serviço Web portáveis e interoperáveis".

As tecnologias para comunicação são as seguintes:

- Protocolos de Internet – A plataforma J2EE suporta protocolos comuns da Internet, tais como, TCP/IP, HTTP, SSL, entre outros, estes protocolos permitem a comunicação entre componentes e seus clientes.
- Invocação de Métodos Remotos (RMI) – A plataforma J2EE suporta Java RMI, que implementa a invocação de métodos remotos, utilizando classes Java para implementar interfaces de objetos remotos.
- Tecnologias de Mensagem – Esta tecnologia é utilizada para que a plataforma J2EE suporte comunicações assíncronas, como exemplos, pode ser citado a API *Java Message Service* e *Java Mail*.
- Tecnologia de Serviços Web – A plataforma J2EE suporta tecnologias específicas para Serviços Web, possibilitando assim a comunicação com clientes não Java. A plataforma J2EE suporta padrões de Serviços Web, como por exemplo, UDDI e SOAP, utilizando tecnologias como a API Java para RPC Baseado em XML e API Java para Registros XML.

3.3 API Java para RPC Baseado em XML – (JAX-RPC)

A API Java para RPC baseado em XML (JAX-RPC) suporta chamadas remotas a procedimentos (RPC), utilizando um protocolo baseado em XML nas plataformas Java e J2EE. A API JAX-RPC possibilita o desenvolvimento de Serviços Web baseados em SOAP, portáveis e interoperáveis.

A API JAX-RPC é utilizada para o desenvolvimento de aplicações finais, ou seja, a aplicação serviço do Serviço Web baseada em SOAP, juntamente com sua correspondente descrição WSDL, e clientes.

Uma implementação de Serviço Web baseado em JAX-RPC pode interagir com clientes que não são baseados em Java, da mesma forma que, clientes baseados em JAX-RPC podem se comunicar com implementações de Serviços Web não baseados em Java (SINGH, 2004).

Em um cenário típico de implementação de Serviços Web a utilização de JAX-RPC reduz a complexidade no desenvolvimento das seguintes funcionalidades:

- Criação de requisições e respostas SOAP padrões.
- Conversão dos parâmetros conhecidos como “marshalling” e “unmarshalling” entre outros detalhes específicos de tempo de execução e montagem.
- Livra o programador da tarefa de criar e converter mensagens SOAP, fornecendo estas funcionalidades em bibliotecas e ferramentas.

A JAX-RPC também define padrões de mapeamento entre WSDL/XML e Java, possibilitando um grande conjunto de tipos. Entretanto, desenvolvedores podem utilizar tipos que não tenham um mapeamento padrão. JAX-RPC fornece várias APIs para um amplo mapeamento de tipos, que pode ser utilizado pelo desenvolvedor para tipos não mapeados (SINGH, 2004).

Uma aplicação cliente pode efetuar uma requisição ao Serviço Web de três maneiras descritas a seguir:

a) Invocação de métodos em *stubs* gerados - Baseado no conteúdo de uma descrição WSDL de um Serviço Web, ferramentas podem ser utilizadas para gerar os *stubs*. Estes *stubs*

possuem toda a informação referente ao Serviço Web e sua localização. A partir destes stubs as aplicações cliente invocam métodos remotos contidos na aplicação serviço.

b) Utilizando *proxy* dinâmico – Um *proxy* dinâmico suporta uma aplicação serviço. Quando este modelo é utilizado não há necessidade de criar *stubs* específicos para as aplicações serviço para o cliente.

c) Interface de Invocação Dinâmica (DII) – Neste modelo, operações no serviço alvo são acessadas dinamicamente baseadas em um modelo da descrição WSDL do serviço, em memória.

Independente do modelo usado, as requisições de aplicações clientes passam através de uma JAX-RPC em tempo de execução do lado do cliente. O processo de tempo de execução mapeia os tipos Java da requisição para XML e forma uma mensagem SOAP correspondente para a requisição. Ela então envia a mensagem SOAP através da rede para o servidor.

No lado do servidor, a JAX-RPC em tempo de execução recebe a mensagem SOAP da requisição. O tempo de execução do servidor aplica o mapeamento reverso de XML para Java, e então mapeia a requisição para o correspondente método Java chamado, junto com seus parâmetros.

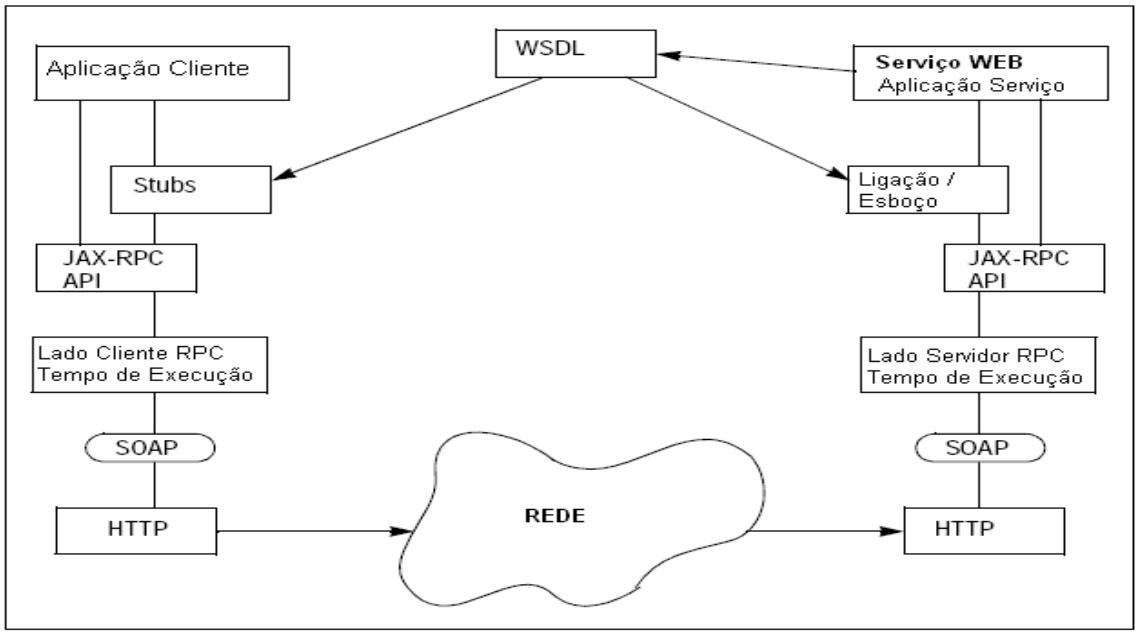


Figura 7 – Arquitetura JAX-RPC (SINGH, 2004)

Esta comunicação descrita pode ser realizada com clientes ou serviços não Java, desde que ambas as partes utilizem o protocolo SOAP. O programador não precisa se preocupar com os detalhes deste processo, a maioria dos detalhes estão ocultos.

Na Figura 7 descreve-se uma visão de alto nível da implementação da arquitetura JAX-RPC, onde tem-se todo o processo de comunicação entre uma aplicação cliente e um Serviço Web.

A arquitetura JAX-RPC suporta três modos de operação, descritos abaixo:

1) Solicitação-Resposta no modo Síncrono – Depois que um método remoto é invocado, o serviço de linha de execução do cliente é bloqueado até que um valor ou exceção é retornado.

2) Unidirecional modo RPC – Depois que um método remoto é invocado, a linha de execução do cliente não é bloqueada e continua o processamento. Nenhum valor de retorno ou exceção é esperado nesta chamada.

3) Sem bloqueio no modo de invocação RPC – O cliente invoca um método e continua em sua linha de execução sem bloqueio. Posteriormente, o cliente processa a resposta do método remoto, realizando um recebimento de chamada bloqueante.

No Quadro 5 é apresentado um exemplo de interface de um serviço JAX-RPC para um Serviço de Prontuário de Pacientes:

```
public interface ProntuarioWSSEI extends java.rmi.Remote {
      public String enviaDadosPaciente(java.lang.String docXML)
                  throws java.rmi.RemoteException;
}
```

Quadro 5 – JAX-RPC Fragmento de uma Interface

No Quadro 6 descreve-se a implementação do Serviço descrito na interface da Quadro 5:

```
public class ProntuarioWSImpl implements ProntuarioWSSEI {
    private LerDadosPaciente ler;
    public String enviaDadosPaciente(java.lang.String docXML)    {
        ler = new LerDadosPaciente(docXML);
        return ler.insereDados();
     }
 }
```

Quadro 6 – JAX-RPC Fragmento de uma Implementação de Interface

No Quadro 7 descreve um cliente utilizando RPC para acessar o Serviço Web de Prontuário de Pacientes:

```
Context ic = new InitialContext();
Service svc = (Service)
   ic.lookup("java:comp/env/service/ProntuarioWS");
ProntuarioWSSEI port =
      (ProntuarioWSSEI) svc.getPort(ProntuarioWSSEI.class);
String resposta = port. enviaDadosPaciente(DadosPacXML);
```

Quadro 7 – Um Cliente Java/J2EE acessando o Serviço Web de Prontuário de Pacientes

Com estes exemplos pode ser visto que o desenvolvedor tem que escrever pouco código para configuração e montagem das aplicações. JAX-RPC cuida de detalhes de implementação, como criar uma requisição SOAP, manipular a resposta SOAP, entre outras, por meio disso aliviando o desenvolvedor destas complexidades.

3.4 API Java para Registros XML (JAXR)

Segundo SINGH (2004):

> "A API Java para Registros XML (JAXR), é uma API para acessar registros de negócios, com uma arquitetura flexível com suporte a UDDI e outras especificações de registro como ebXML. Um cliente JAXR, que pode ser uma aplicação Java isolada ou um componente J2EE, utiliza uma implementação da API JAXR fornecida por um provedor JAXR para acesso a um registro de negócios".

Um provedor JAXR consiste de duas partes: um JAXR específico para o registro, que fornece uma implementação da API específica para o registro, e um JAXR fornecedor de conexão, que implementa as funcionalidades da API que são independentes do tipo de registro.

Um JAXR fornecedor de conexão oculta do cliente os detalhes específicos de registros de fornecedores.

Um provedor de registro específico conecta no fornecedor de conexão, e age sobre requisições e respostas entre o cliente e o registro alvo. O fornecedor de registro especifico, converte a requisição do cliente em um formato reconhecível pelo registro alvo e envia a requisição para o fornecedor do registro utilizando protocolo de registro especifico (SINGH, 2004).

O provedor de registro específico, também é responsável pela conversão de respostas do fornecedor de registros para um formato especifico de resposta JAXR, e então passa a resposta para o cliente.

Na Figura 8 apresenta-se a interligação da arquitetura JAXR com registros e clientes.

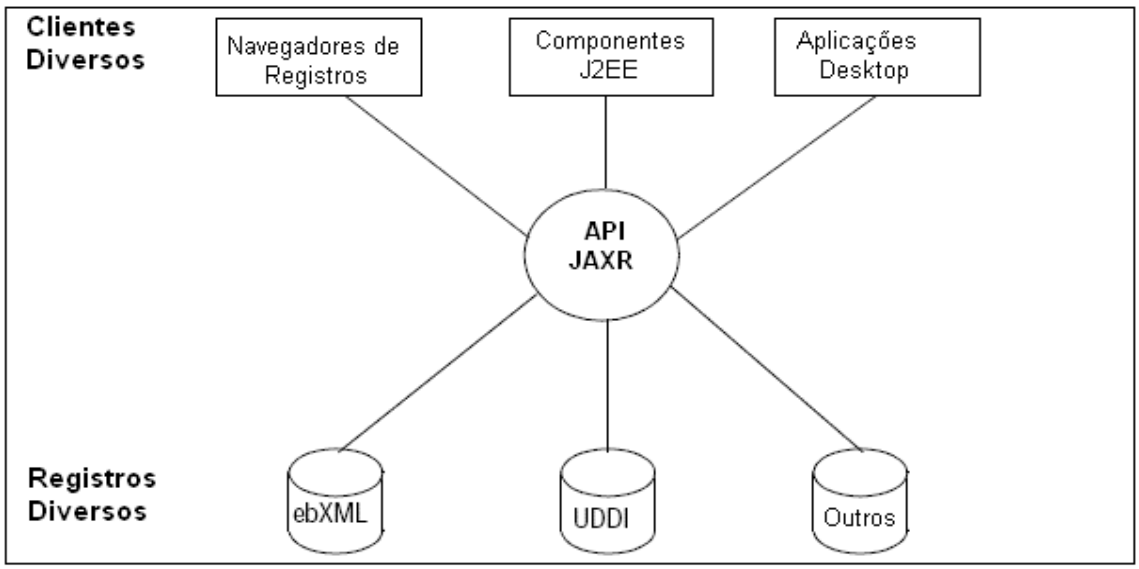


Figura 8 – Arquitetura JAXR (SINGH, 2004)

3.5 API Java para Processamento XML - (JAXP).

A API Java para processamento de XML é um conjunto de APIs leves, independentes de fornecedores para conversão ou processamento de documentos XML (SINGH, 2004).

XML é a linguagem comum que viabiliza Serviços Web, assim torna-se necessário, mecanismos de conversão e de processamento das mensagens, ou seja, documentos XML trocados entre os Serviços Web.

A API JAXP não é nova na plataforma J2EE, sua implementação já estava presente em versões anteriores de J2EE e J2SE. Na versão 1.4 de J2EE foi adicionado suporte a esquemas XML (XSD) (SINGH,2004).

A API JAXP possui sua própria referência de implementação, porém permite conversão entre implementações de outros fornecedores para ser portável. JAXP utiliza sua própria implementação se uma outra implementação não for fornecida.

Na Figura 9 descreve-se a utilização da API JAXP para abstração de conversão de documentos XML de aplicações de usuários. Onde estas aplicações se comunicam com a API JAXP e esta fica responsável pela comunicação com a implementação de Referência e Fornecedor.

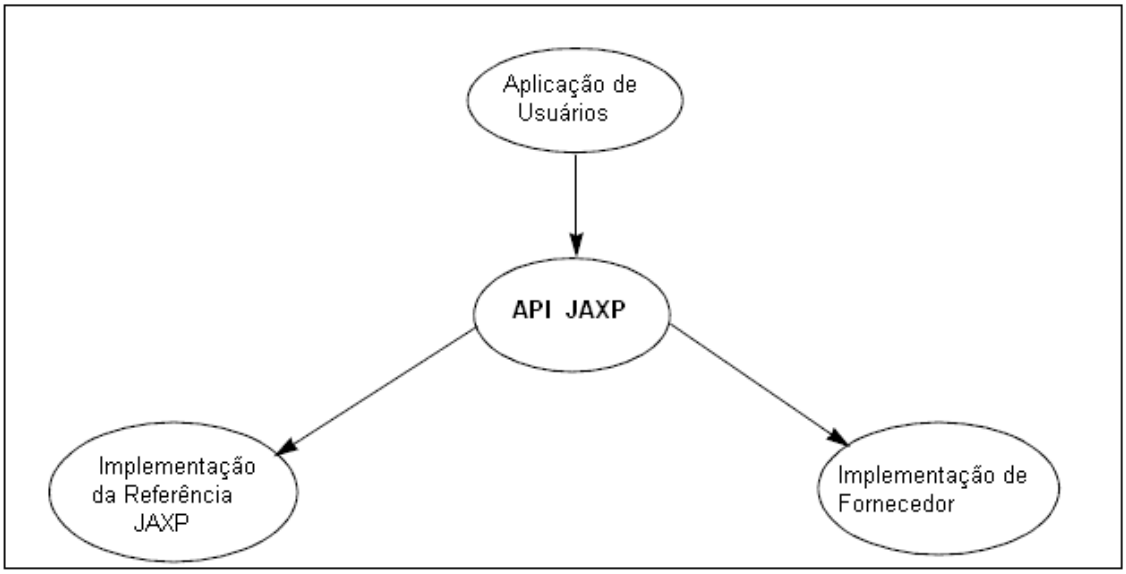


Figura 9 – Utilizando JAXP para abstração da conversão de documentos XML

(SINGH, 2004)

JAXP processa documentos XML utilizando os modelos SAX *(Simple API for XML)* ou DOM *(Document Object Model)*, permite utilização do XSLT *(eXtensible Stylesheet Language Transformation)*, um método de conversão de um formato XML para outro.

O modelo de processamento SAX, processa XML de forma serial, convertendo os elementos de um documento XML em uma série de eventos. Cada elemento em particular cria um evento e um único evento representa várias partes de um documento.

O modelo DOM cria uma árvore dos documentos contidos no documento XML, deste modo, requerendo mais memória para armazenar a árvore. Esta funcionalidade permite acesso aleatório ao documento e permite divisão do documento em fragmentos.

O modelo SAX é rápido por permitir acesso em seqüência e ser necessário pouco espaço de memória para armazenamento.

3.6 Tecnologias de Serviços Web integradas na plataforma J2EE

Na plataforma J2EE um Serviço Web pode ser implementado de duas maneiras, utilizando componentes JAX-RPC ou EJB.

Utilizando JAX-RPC como aplicação serviço, a implementação consiste de uma classe em um *Container* Web. O serviço adere aos parâmetros do container, baseando-se no ciclo de vida de um *Servlet* e requisitos de funcionamento.

Utilizando EJB como aplicação serviço, a implementação é um componente *Session Bean* sem *status* em *Container* EJB. O serviço adere aos parâmetros do ciclo de vida do *container* e requisitos de funcionamento.

Em ambos os cenários, o Serviço Web torna-se portável com a definição de um componente porta, que fornece uma visualização exterior da implementação do Serviço Web.

Um componente porta consiste de:

- Um documento WSDL descrevendo o serviço que o cliente pode usar.
- A interface da implementação do serviço que define os métodos do Serviço Web que estarão disponíveis para o cliente.
- Uma *bean* implementando a lógica dos métodos definidos na interface de implementação do serviço. Esta implementação pode ser tanto uma classe Java em um *Container* Web, quanto um *Session bean* sem *status* em um *Container* EJB (SINGH, 2004).

Interfaces específicas do serviço de *container*, criadas pelo *container* EJB, fornecem *stubs* estáticos e *proxys* dinâmicos para todas as portas. Um cliente de um Serviço Web da plataforma J2EE pode ser um outro Serviço Web, um Componente J2EE ou uma aplicação isolada. Ele não requer que um cliente seja um Serviço Web ou uma aplicação implementada em Java.

A seguir é descrita a seqüência para que um cliente J2EE possa acessar um Serviço Web qualquer:

1) Por ser um cliente, utiliza JNDI para procurar por um serviço.

2) Depois de encontrado, acessa a porta do Serviço Web, utilizando métodos definidos na interface `javax.xml.rpc.Service`.

3) O cliente acessa as funcionalidades do serviço utilizando a interface da aplicação serviço.

4) Um cliente que é um componente J2EE necessita apenas considerar que o Servidor Web é sem *status*. Isto significa que o cliente não necessitará manter estados entre sucessivas invocações ao serviço.

3.7 Fluxo de uma chamada a um Serviço Web J2EE

No cenário de Serviços Web, um cliente faz uma requisição para um serviço em particular e o serviço após processar a requisição, envia uma resposta para o cliente, completando a requisição.

Quando o cliente e o servidor são implementados no ambiente Java, o cliente cria as chamadas para o serviço invocando um método Java, que é passado com seus parâmetros para a JAX-RPC em tempo de execução do lado do cliente. Com a chamada ao método, o cliente está invocando uma operação no servidor.

Estas operações representam um dos diferentes serviços de interesse do cliente. A JAX-RPC em tempo de execução mapeia os tipos Java para tipos XML padrões e forma uma mensagem SOAP que encapsula a chamada do método e seus parâmetros.

O tempo de execução então passa a mensagem SOAP através do manipulador SOAP, se existir algum, então passa para o serviço de porta do lado do servidor.

As chamadas de aplicações cliente atingem o serviço através de uma porta, desde que a porta forneça acesso sobre um formato de dados e protocolo específico sobre a rede com uma interface de aplicação serviço consistindo de um endereço e uma porta.

Antes da porta passar a requisição para a aplicação serviço, ela assegura que o *container* J2EE está de acordo com serviços declarados para a requisição SOAP

Depois do manipulador operar sobre a mensagem SOAP a mensagem é passada para a aplicação serviço.

O *container* J2EE extrai as chamadas de métodos invocadas pelo cliente e ajusta com os parâmetros para a chamada, realiza algum mapeamento de XML para Java necessário, e passa o método para a implementação da interface do Serviço Web para realizar o processamento.

Uma resposta enviada do servidor segue um processo similar.

O comportamento do lado do cliente é como descrito acima apenas quando o cliente é desenvolvido em ambiente Java, quando for desenvolvido em outra plataforma, ele deverá refletir as ações da plataforma específica (SINGH, 2004).

3.8 Outra tecnologia para ligação XML - JAXB

A arquitetura Java para ligação XML (JAXB), não é uma tecnologia essencial para implementação de Serviços Web, porém é uma API muito útil, utilizada para padronizar a representação de um documento XML como um objeto em memória.

Quando trocamos documentos entre aplicações, os documentos XML devem seguir algumas estruturas para que as partes envolvidas possam compreender o conteúdo dos documentos.

Utilizando esquema XML (XSD) é possível, utilizando sintaxe XML descrever o relacionamento entre elementos, atributos e entidades em um documento XML. O objetivo de um esquema XML é definir um conjunto de documentos XML que deverão seguir um conjunto estrutural de regras e conteúdo de dados.

O desenvolvedor pode utilizar os modelos DOM ou SAX para ligar XML, mas isso é facilitado se as várias partes do documento XML estão mapeadas ou limitadas em objetos em memória que representam verdadeiramente o documento XML pretendido, através da definição do esquema XSD.

Para utilizar estes objetos o desenvolvedor deve acessar o esquema XML como parte de sua lógica.

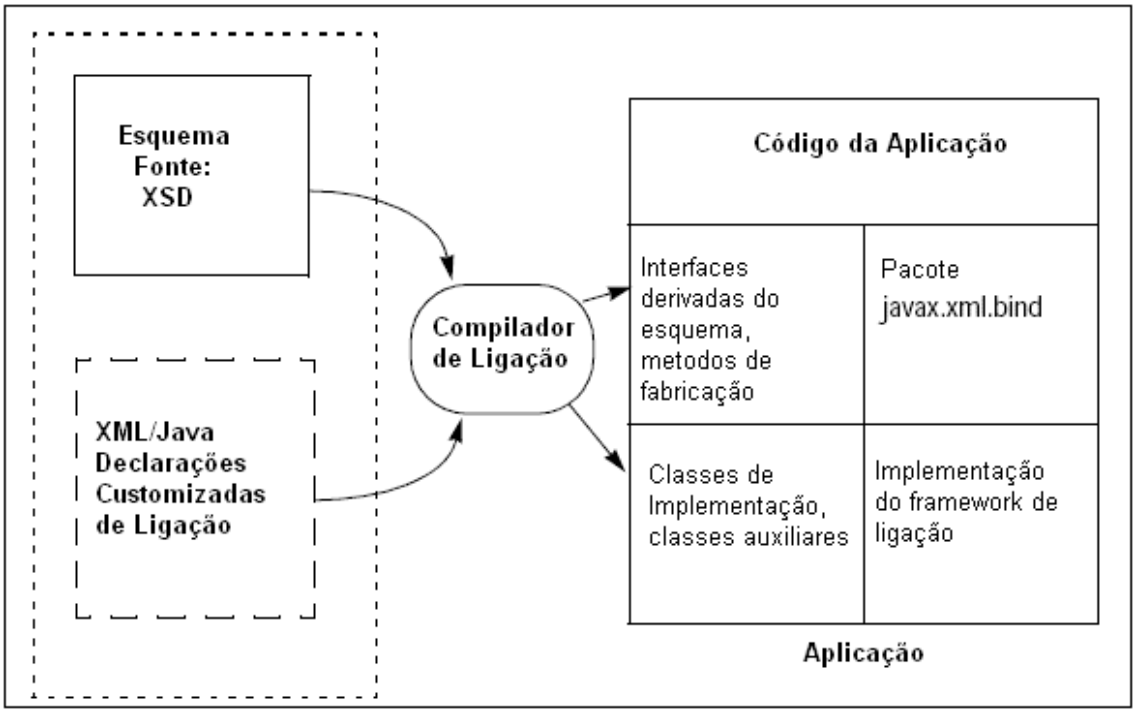


Figura 10 – Arquitetura JAXB (SINGH, 2004)

Segundo SUN (2004), a arquitetura básica da JAXB consiste de três componentes como segue:

- Um compilador de ligação que cria classes Java a partir de um dado esquema. Tipos de dados complexos dentro do esquema são mapeados para classes separadas, enquanto tipos simples são definidos como campos dentro de uma classe. O desenvolvedor utiliza métodos `get` e `set` para acessar e modificar o conteúdo do objeto.
- Um framework de ligação que fornece serviços em tempo de execução para converter arquivos XML em objetos Java chamado de *unmarshalling* e o processo oposto chamado de *marshalling* e validação para validar um dado documento XML com um respectivo esquema XML.
- Uma linguagem de ligação que descreve a ligação do esquema para classes Java. Esta linguagem possibilita ao desenvolvedor sobrepor às regras de ligação padrão, com isso possibilitando ao desenvolvedor customizar as classes que são criadas pelo compilador de ligação.

Na Figura 10 mostra-se a arquitetura JAXB, descrevendo um esquema de seu inter-relacionamento com aplicações. Tem-se os XSD e Declarações de Ligação se comunicando com um Compilador de Ligação que gera as classes necessárias.

4 IMPLEMENTAÇÃO DO SERVIÇO WEB

Para demonstrar a utilização da Plataforma de desenvolvimento Java/J2EE na implementação de Serviços Web, foi necessário a escolha de um cenário onde esta tecnologia pudesse ser utilizada de forma a facilitar a troca de informações entre entidades com algum interesse em comum.

Este capítulo mostrará os detalhes da aplicação escolhia, ferramentas utilizadas para o desenvolvimento e como foi realizado o desenvolvimento do Serviço Web.

4.1 Aplicação Escolhida

Os serviços de saúde possuem grandes obstáculos para a centralização e compartilhamento de informações referentes ao ciclo de vida de um paciente.

Um paciente vai a um hospital realizar alguma consulta, porém aquelas informações sobre o paciente ficam apenas no sistema de informação interno do hospital. Surge a grande necessidade de compartilhar estas informações com outros hospitais, clínicas e laboratórios.

Um problema enfrentado é a incompatibilidade de comunicação entre os sistemas de informação já existentes neste cenário. Algumas entidades já possuem seu sistema de informação funcionando corretamente e não pretende mudar de linguagem ou plataforma.

Para solucionar este problema de incompatibilidade na troca de informação foi projetado um Serviço Web, um ponto centralizado para armazenar e distribuir estas informações entre diversas entidades interessadas.

O projeto foi desenvolvido com base em especificações já existentes, com dados essenciais para que esta troca de informação seja útil a todos interessados.

4.1.1 Conjunto Essencial de Dados

Em outubro de 1999, um comitê formado por integrantes de diversas instituições de saúde, validou um conjunto essencial de dados do prontuário para integração da informação na saúde, sob o código SOP 001/98 (LIRA,1999).

Este processo envolveu diversas instituições de saúde que participaram propondo seus pontos de vista sobre este conjunto essencial de dados.

A recomendação final SOP 001/98 contou com representantes das seguintes instituições: Hospital das Clínicas da Faculdade de Medicina da USP-HCFMUSP, Hospital das Clínicas de Porto Alegre - HCPA, Companhia de Processamento de Dados de Porto Alegre – PROCEMPA, Hospital Pronto Socorro de Porto Alegre, Hospital Moinhos do Vento, INFOSAUDE e Hospital Mão de Deus (LIRA,1999).

O objetivo desta proposta foi de apresentar um conjunto essencial de dados clínicos que visam suportar a assistência aos pacientes, não foi voltado a dados administrativos ou de faturamento, nem a qualquer tipo de especialização para programas ou especialidades específicas.

Segundo LIRA (1999), a abrangência da proposta é para que independente do paciente ter sido internado, atendido em ambulatório, ou ainda, por algum programa especial, exista um conjunto essencial de dados a partir do qual possam ser construídas as especializações para as demais áreas.

Esta proposta tem por objetivo, em longo prazo, entender este conjunto essencial de dados para acomodar mais dados clínicos, laudos médicos e até mesmo imagens. Para abranger em sua totalidade os dados do paciente.

A recomendação abrange os seguintes dados: Informações administrativas e demográficas, como nome, endereço, dados familiares, entre outros. Dados administrativos do prestador de assistência e fonte pagadora. Dados clínicos considerados relevantes, de preenchimento opcional, como alergias ou reações adversas, doenças crônicas pré-existentes, que determinado paciente possa apresentar.

Um acompanhamento de eventos e atendimentos que possam ocorrer durante a vida do paciente, nesta proposta um evento ou atendimento, ocorre apenas quando existe um contato entre um paciente e um profissional da saúde.

Incluem ainda dados referentes a exames laboratoriais realizados pelos pacientes.

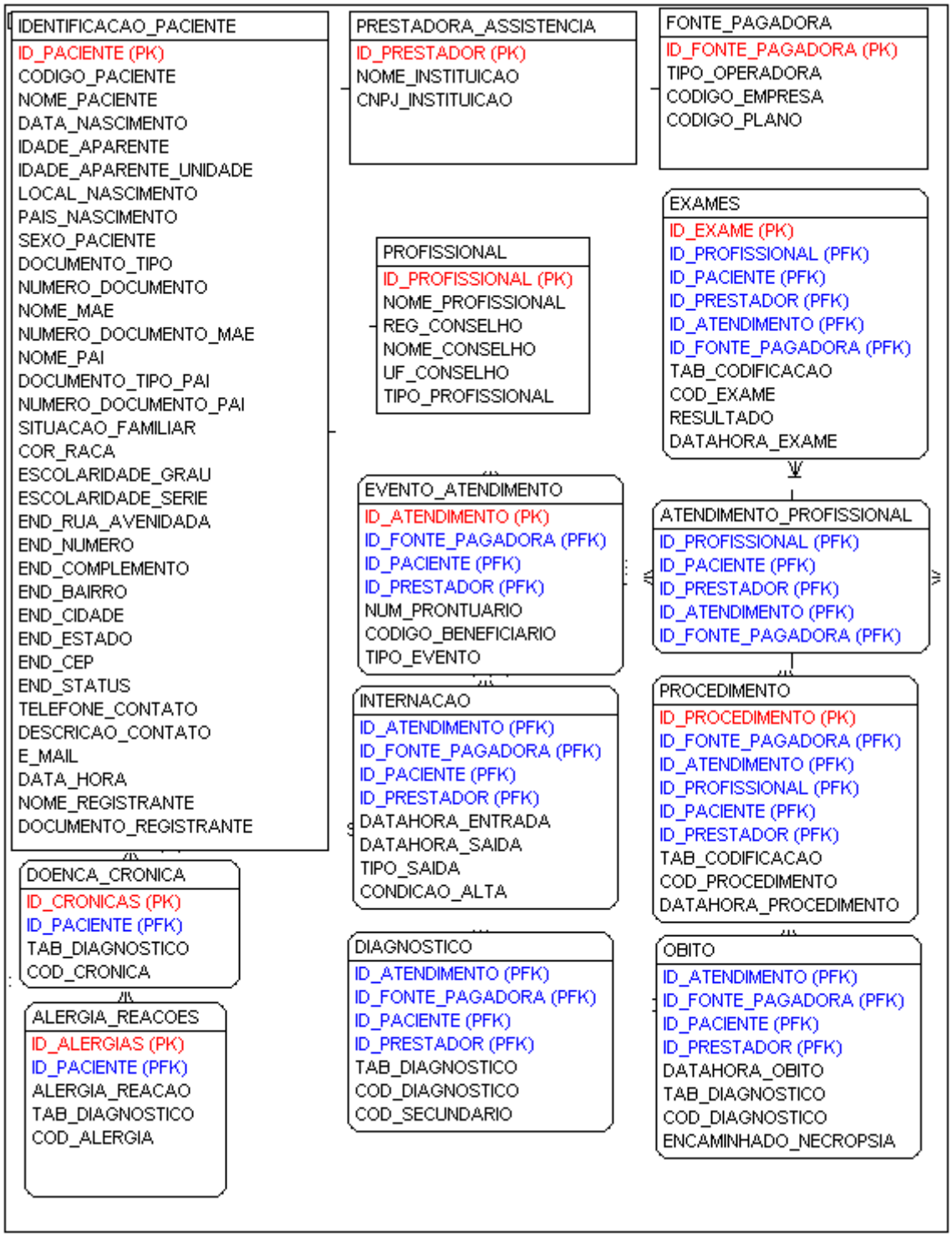


Figura 11 – Fragmentos do Modelo da Base de Dados – ProntuarioWS

Na Figura 11 apresenta-se alguns fragmentos do modelo da base de dados gerada a partir da análise de recomendação. Esta base de dados será utilizada pelo Serviço Web para centralizar as informações referentes aos pacientes e seus eventos e atendimentos.

O modelo completo descrevendo a base de dados utilizada neste projeto, com seus respectivos inter-relacionamentos encontra-se no Apêndice A.

4.2 Ferramentas utilizadas na implementação do Serviço e Servidor Web

A implementação deste serviço buscou desenvolver um projeto em que todos os softwares utilizados fossem de distribuição livre, para que não houvesse a necessidade de gastos com licença de softwares aplicativos ou mesmo com sistemas operacionais.

Os softwares utilizados juntamente com algumas especificações técnicas a respeito serão apresentados nas seções deste capítulo.

4.2.1 Sistema Operacional Red Hat Linux 9

O sistema operacional Red Hat Linux 9, distribuição de 2003, foi escolhido por ser uma distribuição popular, madura e estável do Linux.

O usuário não precisa pagar para adquirir esta versão do Linux, terá gastos apenas com suporte se for preciso.

Um outro fator que contribui para escolha desta distribuição do Linux é sua compatibilidade com ferramentas de desenvolvimento como Net Beans e a Plataforma de Desenvolvimento J2EE.

Abaixo seguem algumas especificações de Hardware necessárias para instalação desta versão do Red Hat 9, de acordo com suas especificações técnicas, disponível no CD de instalação :

- CPU - No mínimo pertencente à classe Pentium, com capacidade de processamento 400MHZ Pentium II, ou melhor.

- Espaço em disco - Para uma instalação completa da distribuição ocupa-se 5.0 GB, porém para outras instalações como apenas Servidor, ocuparia 850 MB.
- Memória – É recomendado que utilize 190 MB no modo gráfico.

A empresa fornecedora deste produto é a Red Hat, Inc.

4.2.2 Java e J2EE

Para esta implementação foi necessário instalar o *kit* de desenvolvimento Java 2 Platform, Standard Edition (J2SE). A versão utilizada nesta implementação foi a J2SE v.1.4.2_8 SDK.

A plataforma de desenvolvimento *Java 2 Platform, Enterprise Edition* (J2EE). A versão utilizada foi a J2EE 1.4 SDK.

O kit de desenvolvimento de Serviços Web *Java Web Services Developer Pack* "Java WSDP", na versão 1.5.

A empresa fornecedora destes produtos é a Sun Microsystems, Inc.

4.2.3 Sun Java System Application Server

Para servidor de aplicação foi utilizado *Sun Java System Application Server Platform Edition 8.1 2005Q2*. Esta distribuição é livre apenas para desenvolvimento, para redistribuição torna-se necessário adquirir licença.

A empresa fornecedora deste produto é a Sun Microsystems, Inc.

4.2.4 NetBeans IDE 4.1

Para auxiliar no desenvolvimento do Serviço Web foi utilizada a ferramenta IDE *NetBeans Platform 4.1*. Que fornece suporte ao desenvolvimento de Serviços Web utilizando a plataforma J2EE.

Fornecedor: NetBeans.org.

4.3 Abordagem da Implementação realizada

Nesta seção serão descritas algumas das características do Serviço Web ProntuarioWS implementado, e as abordagens do desenvolvimento, adotadas nas principais fases da implementação.

4.3.1 Interface que será disponibilizada para os clientes.

Para o desenvolvimento da definição da interface do Serviço Web em plataforma J2EE foi possível a escolha entre duas abordagens:

A abordagem Java para WSDL, na qual o desenvolvimento inicia-se com um conjunto de interfaces Java para Serviços Web, a partir da qual, cria-se o documento WSDL, com a descrição do Serviço que tornará possível a comunicação com as aplicações cliente.

Outra abordagem seria a WSDL para Java, na qual o desenvolvimento inicia-se com a criação do documento WSDL, descrevendo os detalhes da interface do Serviço Web, e utilizando estas informações é possível gerar as interfaces Java correspondentes.

A abordagem de desenvolvimento escolhida neste trabalho foi o desenvolvimento Java para WSDL.

Um das características desta abordagem é a existência de diversas ferramentas de auxilio ao desenvolvimento, que geram facilmente o documento WSDL a partir das interfaces

Java, por exemplo a NetBeans IDE. No entanto, a utilização destas ferramentas faz com que o desenvolvedor perca o controle sobre alguns aspectos de criação do documento WSDL.

Na abordagem WSDL para Java, é possível um maior controle sobre as interfaces que serão desenvolvidas, dando maior flexibilidade ao processo de desenvolvimento, Entretanto nesta abordagem é exigido um bom conhecimento do desenvolvedor sobre especificação WSDL e WS-I, para tirar vantagem destas características.

Outra característica importante na escolha da interface a ser utilizada para o desenvolvimento do Serviço Web, é o tipo; a plataforma J2EE possibilita a escolha entre JAX-RPC e EJB.

O Serviço JAX-RPC, também conhecido como serviço da Camada Web, é mais utilizado quando o processamento ocorre todo sobre a Camada Web. Por sua vez o EJB é utilizado quando o processamento ocorre sobre a camada de regras de negócio.

Neste projeto optou-se pela escolha do modelo JAX-RPC, por não possuir uma regra de negócio já definida a seguir e pelo fato do processamento ser feito todo na Camada Web.

Como pode ser visto no Quadro 8 em que é mostrado fragmentos da interface do Serviço Web implementado neste projeto, a operação `enviaDadosPaciente` recebe uma *String* contendo um arquivo XML com os dados dos Pacientes como parâmetro. Esta *string* deve obedecer ao padrão estabelecido no arquivo `Identificacao_Paciente.xsd`.

A operação `buscarDadosPaciente`, recebe o Código do Paciente como parâmetro `idPaciente`, e retorna uma string com o arquivo XML com os Dados do Paciente, também seguindo o padrão do documento XSD.

```
package org.anderson.prontuario;
public interface ProntuarioWSSEI extends java.rmi.Remote {
    /** Operação do Serviço Web **/
  public String enviaDadosPaciente( java.lang.String docXML)
                        throws java.rmi.RemoteException;
   /**  Operação do Serviço Web**/
  public String buscarDadosPaciente(long idPaciente)
                        throws java.rmi.RemoteException;
      // Outras operações do Serviço Web
}
```

Quadro 8 - Fragmento da interface Java do Serviço Web ProntuarioWS

Com base na Interface Java é criado o documento WSDL, como pode ser visto no Quadro 9, contendo todos os métodos que estarão disponíveis no Serviço Web ProntuarioWS para os clientes.

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="ProntuarioWS"
      targetNamespace="urn:ProntuarioWS/wsdl"
      xmlns:tns="urn:ProntuarioWS/wsdl"
      xmlns="http://schemas.xmlsoap.org/wsdl/"
      xmlns:xsd="http://www.w3.org/2001/XMLSchema"
      xmlns:ns2="urn:ProntuarioWS/types"
      xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
<types>
      <schema targetNamespace="urn:ProntuarioWS/types"
                  xmlns:tns="urn:ProntuarioWS/types" xmlns:soap11-
                        enc="http://schemas.xmlsoap.org/soap/encoding/"

      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns="http://www.w3.org/2001/XMLSchema">
            ……
            <complexType name="buscarDadosPaciente">
                        <sequence>
                              <element name="long_1" type="long"/>
                        </sequence>
            </complexType>
                  <complexType name="buscarDadosPacienteResponse">
                        <sequence>
                          <element name="result" type="string"
                  nillable="true"/>
                        </sequence>
            </complexType>
            ……
            <element name="buscarDadosPaciente"
                        type="tns:buscarDadosPaciente"/>
                  <element name="buscarDadosPacienteResponse"
                        type="tns:buscarDadosPacienteResponse"/>
            ……
      </schema>
</types>
……
<message name="ProntuarioWSSEI_buscarDadosPaciente">
      <part name="parameters"      element="ns2:buscarDadosPaciente"/>
</message>
<message name="ProntuarioWSSEI_buscarDadosPacienteResponse">
      <part name="result" element="ns2:buscarDadosPacienteResponse"/>
</message>
……
<portType name="ProntuarioWSSEI">
……
      <operation name="buscarDadosPaciente">
                  <input
message="tns:ProntuarioWSSEI_buscarDadosPaciente"/>
                  <output message=

      "tns:ProntuarioWSSEI_buscarDadosPacienteResponse"/>
      </operation>
……
```

```
</portType>
<binding name="ProntuarioWSSEIBinding" type="tns:ProntuarioWSSEI">
……
      <operation name="buscarDadosPaciente">
                  <soap:operation soapAction=""/>
             <input>
                        <soap:body use="literal"/>
             </input>
                  <output>
                        <soap:body use="literal"/>
                  </output>
      </operation>
……
</binding>
<service name="ProntuarioWS">
      <port name="ProntuarioWSSEIPort"
            binding="tns:ProntuarioWSSEIBinding">
                  <soap:address location="REPLACE_WITH_ACTUAL_URL"/>
      </port>
</service>
</definitions>
```

Quadro 9 - Fragmento do documento WSDL do Serviço Web ProntuarioWS.

4.3.2 Padronização de Informações através de esquemas XML (XSD)

Para viabilizar a troca de informações entre o Serviço Web e os clientes, de forma padronizada, foram criados esquemas XML.

O objetivo de um esquema XML é definir um conjunto de documentos XML que deverá seguir um conjunto estrutural de regras e conteúdo de dados. Desta forma, expressa uma forma de compartilhar vocábulos, permitindo aos computadores seguirem regras criadas por pessoas.

Os esquemas XML são a base para que a informação possa fluir neste Serviço Web implementado, pois o Serviço Web só aceitará as mensagens na qual o documento XML seja validado com seu respectivo esquema XSD, e as respostas geradas pelo Serviço Web também seguirão o mesmo padrão, estarão em um formato compatível com o esquema XSD correspondente.

O Serviço Web ProntuarioWS, utiliza os seguintes esquemas XSD:

- `Indetificação_Paciente.xsd` – Padronizado baseado no conjunto essencial de dados. Descreve quais informações devem ser enviadas ao serviço, para que a inclusão de um paciente possa ser realizada com sucesso. Também

especifica o padrão que o serviço utilizará para as respostas a solicitação de informações sobre pacientes.

- `Prestadora_Assistencia.xsd` – Também padronizado com base no conjunto essencial de dados. Estabelece as informações necessárias a respeito da Instituição Prestadora da Assistência a Saúde.
- `Fonte_Pagadora.xsd` – Padroniza as informações a respeito da Instituição Pagadora do Evento ou Procedimento realizado pelo profissional ou instituição de saúde.
- `Profissional.xsd` – Padroniza os dados essências, a respeito do profissional da saúde responsável pelo evento ou atendimento do paciente.
- `EventoAtendimento.xsd` – Padroniza os dados essências a respeito de interações entre um paciente e um profissional ou equipe de saúde.

```
<?xml version="1.0" encoding="UTF-8"?>
      <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
      <xs:element name="FONTE">
      <xs:annotation>
      <xs:documentation> Comentario </xs:documentation>
      </xs:annotation>
            <xs:complexType>
                  <xs:sequence maxOccurs="unbounded">
                     <xs:element name="ID_FONTE"
type="xs:unsignedLong"                                    minOccurs="0"/>
                     <xs:element name="TIPO_OPERADORA" type="xs:int"
                              minOccurs="0"/>
                        <xs:element name="CODIGO_EMPRESA"
type="xs:unsignedLong" minOccurs="0"/>
                        <xs:element name="CODIGO_PLANO"
type="xs:integer"                                  minOccurs="0"/>
                  </xs:sequence>
            </xs:complexType>
      </xs:element>
</xs:schema>
```

Quadro 10 - Documento Fonte_Pagadora.xsd.

No Quadro 10, descreve-se o conteúdo do documento `Fonte_Pagadora.xsd`, com suas regras e dados essências que um documento XML deve seguir para ser aceito pela operação `buscarDadosPrestadora` do Serviço ProntuarioWS.

Este documento define os elementos ID_FONTE, TIPO_OPERADORA, CODIGO_EMPRESA, CODIGO_PLANO, com seus respectivos tipos e ocorrências. Por

exemplo, o CODIGO_PLANO é um elemento do tipo Inteiro, que pode não ocorrer, no arquivo XML.

No Apêndice B estão todos estes documentos XSDs na íntegra.

4.3.3 Classe de Implementação da Interface do Serviço ProntuarioWS

Para cada operação de envio e busca de dados, contidas na interface do Serviço Web, existe uma respectiva classe Java, responsável pela conversão e tratamento das informações.

Os objetos destes tipos são instanciados a partir da classe responsável pela implementação da interface do Serviço Web ProntuarioWS, como pode ser visto no Quadro 11 onde, tem-se o fragmento da Classe de implementação do Serviço com seus respectivos métodos `enviaDadosPaciente` e `buscarDadosPaciente`,.

O método `enviaDadosPaciente`, recebe a *string* docXML como parâmetro, instancia o objeto `InserirPaciente`, passando a String docXML como parâmetro, chama o método insereDados, que realiza os procedimentos necessários para a leitura e gravação dos dados contidos no documento XML, na base de dados do Serviço.

O método `buscarDadosPaciente`, recebe como parâmetro um `long` contendo o Código do Paciente `idPaciente`, instancia o objeto `buscarPaciente` e chama o método `buscarPaciente` do objeto, passando como parâmetro o idPaciente. Este objeto realiza todos os procedimentos necessários para a busca e geração do arquivo XML, que será retornado para a aplicação Cliente.

```
package org.anderson.prontuario;
import org.anderson.prontuario.paciente.InserirDadosPaciente;
import org.anderson.prontuario.paciente.BuscarDadosPaciente;
.........Importação de outras Classes...........
/**
* Classe Responsável pela Implementação do Serviço Web
**/
public class ProntuarioWSImpl implements ProntuarioWSSEI {
/**
*  Declaração dos Objetos responsáveis pelo tratamento dos parâmetros
* recebidos pelo Serviço ProntuarioWS
**/
```

```
private InserirDadosPaciente inserirPaciente;
private BuscarDadosPaciente buscarPaciente;
/**
*  Métodos responsáveis pela Implementação do Serviço Web.
**/
public String enviaDadosPaciente( java.lang.String docXML) {
                  InserirPaciente = new InserirDadosPaciente(docXML);
                        return inserirPaciente.insereDados();
 }
public String buscarDadosPaciente(long idPaciente) {
                  buscarPaciente = new BuscarDadosPaciente();
                  return buscarPaciente.buscarPaciente(idPaciente);
}
……
// Outras operações do Serviço Web
..….
}
```

Quadro 11 - Fragmento da Classe de Implementação da Interface do Serviço Web.

4.3.4 Geração e Manipulação de documentos XML, utilizando API JAXB

Neste trabalho foi necessário a utilização de uma API que auxilia a manipulação de esquemas XML e documentos XML que seguiam os respectivos padrões estabelecidos nos esquemas XSD.

Optou-se pela utilização da API JAXB, que fornece meios para padronizar a representação de documentos XML como um objeto em memória. Sendo necessário utilizar esquemas XML XSD.

Utilizou-se um compilador de ligação, que criou as classes Java com base nos documentos XML. Os tipos de dados complexos foram mapeados para classes separadas e os tipos simples permaneceram na classe como campos. Foram criados métodos `get` e `set` para acessar e modificar o conteúdo dos objetos, como pode ser visto no Quadro 12, com fragmentos das classes geradas para representação dos esquemas XSD.

Pode ser visto na interface `PACIENTEType.java`, métodos para manipular dados em um objeto que representa o documento XML, como por exemplo, `getCODIGOPACIENTE()` que retorna uma *string* contendo o código do Paciente, e `setCODIGOPACIENTE(java.lang.String value)` que insere um código.

```
//Interface PACIENTE.java
package org.anderson.prontuario.paciente;
public interface PACIENTE extends javax.xml.bind.Element,
                  org.anderson.prontuario.paciente.PACIENTEType
{    }
// interface PACIENTEType.java
package org.anderson.prontuario.paciente;
public interface PACIENTEType {
      java.util.List getIDENTIFICACAOAndENDERECOAndCONTATO();
      public interface ALERGIAREACOES extends javax.xml.bind.Element,
org.anderson.prontuario.paciente.PACIENTEType.ALERGIAREACOESType
      {   }
      public interface ALERGIAREACOESType {
               java.util.List
getIDALERGIASAndIDPACIENTEAndALERGIAREACAO();
               ..........
      }
      .......
      public interface IDENTIFICACAOType{
      java.lang.String getSITUACAOFAMILIAR();
      void setSITUACAOFAMILIAR(java.lang.String value);
      java.lang.String getCODIGOPACIENTE();
      void setCODIGOPACIENTE(java.lang.String value);
      ……
            }
            ..............
}
//Entre outras Classes
```

Quadro 12 - Fragmentos das Classes geradas pelo Compilador de Ligação JAXB

A API JAXB, também fornece um framework de ligação, que foi utilizado para a conversão de arquivos XML em objetos Java, chamado de *unmarshalling* e o oposto chamado de *marshalling* bem como a validação de um dado documento XML com um respectivo esquema XML.

Como pode ser visto no Quadro 13, em que há um exemplo da manipulação da *String* contendo o documento XML, na classe de inserção de dados do paciente no Serviço ProntuarioWS, esta String é transformada em um objeto Java, seta-se então a validação do documento, na existência de algum erro, é enviada para a aplicação cliente uma *String* contendo o numero 1 (um) como caractere inicial seguido da descrição do erro.

A partir destas mensagens de erro o desenvolvedor da aplicação cliente pode verificar onde não foi seguido o padrão do arquivo XSD para geração do arquivo XML e realizar as alterações necessárias.

Se o documento XML for validado serão então lidas todas as informações contidas, através do objeto gerado para este fim, e então serão inseridas estas informações na base de dados do Serviço Web.

Se ocorrer a inserção dos dados com sucesso, será então gerada uma string contendo o numero 0 (zero), como caractere inicial, seguido dos códigos (Identificação na base de dados), das informações enviadas. É importante a aplicação cliente armazenar estes códigos, para que possa consultar estes dados posteriormente, utilizando os códigos gerados.

```
package org.anderson.prontuario.paciente;
import java.io.*;
import javax.xml.transform.stream.*;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Unmarshaller;
import java.sql.ResultSet;
import org.anderson.prontuario.banco.Banco;
........//Importação de outras Classes.........
public class InserirDadosPaciente {
    private JAXBContext jc;
    private Unmarshaller u;
    private PACIENTE pa;
    private PACIENTEType.IDENTIFICACAOType pacIdent;

    …….DECLARAÇÃO DE VARIAVEIS........
public InserirDadosPaciente(String docXML) {
       try
       {
        jc                                                              =
JAXBContext.newInstance("org.anderson.prontuario.paciente");
        u = jc.createUnmarshaller();
        u.setValidating(true);
        xmlStr = new StringBuffer(docXML);
        BD = new Banco();
           …..Tratamento de Exceções…….
  }
  public String insereDados()
  {
       ........Declaração de Algumas Variáveis
      String result = BD.criarConexao();
      if( result != "0") return "1|Erro ao conectar-se a Base de
Dados: Parâmetros de   conexão Incorretos!";
      try {
      pa = (PACIENTE) u.unmarshal
                  (new StreamSource(new
StringReader(xmlStr.toString())));
        List identificacaoList
            = pa.getIDENTIFICACAOAndENDERECOAndCONTATO();

        ........Buscar Código no banco de Dados .........
      for( i = 0; i+1 < identificacaoList.size();i+=6 ) {
          PACIENTE.IDENTIFICACAOType identificação
             = (PACIENTE.IDENTIFICACAOType) identificacaoList.get(i);
             PACIENTE.ENDERECOType   endereco
             = (PACIENTE.ENDERECOType) identificacaoList.get(i+1);
```

```
      .........INSERIR DADOS NO BANCO DE DADOS..............

      SQL = "INSERT INTO IDENTIFICACAO_PACIENTE(" +
                                  "ID_PACIENTE," +  "CODIGO_PACIENTE,"
+..............
      ............ ")VALUES(" +  idPaciente +
                                  " ,'" +
identificacao.getCODIGOPACIENTE() +
                                  "','" + identificacao.getNOMEPACIENTE()
+
                                  "', "  + dataNascimento +
      .............
      }
      .....INCLUIR ALERGIAS A REAÇÕES E DOENÇAS CRONICAS......
      resultado+= idPaciente +"|";
}
```

Quadro 13 - Fragmentos da Classe InserirDadosPaciente

O Serviço Web implementado possibilita a recuperação de dados cadastrados em sua base de Dados. A partir do código de um registro inserido no Banco de Dados é realizada uma consulta e caso seja encontrado o registro é gerado o arquivo XML contendo seus dados e este é enviado para o cliente como retorno do método, o processo oposto ao demonstrado no Quadro 13.

É criado então um documento XML a partir das classes geradas pela API JAXB e então são inseridos os dados referentes ao código da solicitação.

Para tornar a criação destes arquivos XML possível, a API gera uma classe chamada **`ObjectFactory`** que pode ser usada para instanciar objetos dos tipos gerados nas classes anteriormente citadas.

A classe Java `BuscarDadosPaciente` presente no Apêndice C deste trabalho, demonstra todos os detalhes deste processo de criação de XML para a recuperação dos dados de Pacientes da base de dados do Serviço Web ProntuarioWS.

4.3.5 Centralização das Informações em Banco de Dados MySQL

O Serviço Web implementado, possui uma base de dados MySQL, a qual é responsável pelo armazenamento de todos os dados enviados pelas aplicações cliente.

A comunicação com o Servidor MySQL é realizada através da utilização de drivers JDBC, que viabilizam a comunicação de aplicações Java com o Banco de Dados MYSQL.

As classes citadas anteriormente responsáveis pelo processamento das informações, criam um objeto do tipo Banco, que gerência todo processo de comunicação entre a aplicação e o banco de dados, através da passagem de comandos SQL como parâmetros para os métodos de inserção e atualização de dados.

Fragmentos desta classe podem ser vistos no Quadro 14, em que é realizada a conexão com a base de dados do ProntuarioWS.

```
package org.anderson.prontuario.banco;
import java.sql.*;
import com.mysql.jdbc.Driver;
public class Banco {
 ........

    /** Cria uma no instancia de Banco */
    public Banco() {
        this.drivename = "com.mysql.jdbc.Driver";
        this.username = "root";
        this.password = "";
        this.databaseurl = "jdbc:mysql://localhost:3306/PRONTUARIO";
    }
  …….
public String criarConexao(){

            Class.forName("com.mysql.jdbc.Driver").newInstance();
            connection =
DriverManager.getConnection(databaseurl,username,password);
            statement = connection.createStatement();
            return "0";
}
……
public ResultSet executarSQL(String sql){
       try
       {
           rs = statement.executeQuery(sql);
           return rs;
       }
       catch(SQLException exception)
       {   rs = null;
           return rs;
       }
   }
……
}
```

Quadro 14 - Fragmentos da Classe Banco.

O processo de inserção e recuperação de informações na Base de Dados foi testado utilizando a ferramenta NetBeans IDE, que fornece recursos que permitem a criação de aplicações clientes para teste de Serviços Web.

5 CONCLUSÃO

Com base no levantamento e estudo de tecnologias envolvidas na infra-estrutura de Serviços Web e como a plataforma de desenvolvimento J2EE possibilita seu projeto, utilizando-se como base a linguagem de programação Java, constata-se que a plataforma J2EE viabiliza de maneira satisfatória todas as etapas do desenvolvimento, oferecendo APIs e tecnologias consistentes.

O grande número de tecnologias envolvidas na infra-estrutura de Serviços Web, tais como, XML, SOAP, WSDL, UDDI, entre outras, exigem uma grande curvatura de aprendizagem pelo programador. Fator que pode levar a um período extenso de estudo antecedendo o início do desenvolvimento.

A utilização de padrões abertos nas etapas do desenvolvimento do projeto, permitiu grande flexibilidade e robustez das implementações realizadas.

Com a utilização da ferramenta de auxilio de desenvolvimento *NetBeans IDE* foi criada uma aplicação cliente para testar o Serviço. Através dos teste realizados pode se constatar que os métodos implementados neste serviço viabilizam a inserção e recuperação de informações na base de dados do Serviço Web ProntuarioWS de maneira satisfatória.

5.1 Trabalhos Futuros

Como trabalhos futuros para este Serviço deseja-se a complementação de alguns métodos e criação de novos, que possibilitem maior capacidade de pesquisa de informações na base de dados e manutenção das informações contidas na mesma.

E com a complementação de novos projetos a este, espera-se a oportunidade de instalar e configurar este serviço para que Instituições de Saúde possam usufruir do uso desta tecnologia para integração das informações sobre seus Pacientes.

REFERÊNCIAS

CHRISTENSEN, Erik; et al. **Web Services Description Language (WSDL) 1.1.** . [online] Disponível na Internet via WWW. URL: http://www.w3.org/TR/wsdl. Arquivo consultado em 11 de março de 2005.

HENDRICKS, Mack; et al. **Profissional Java Web Services**. Rio de Janeiro: Editora Alta Books, 2002.

LIRA, A. C. O. et al. **SOP 001/98 - Conjunto Essencial de Informações do Prontuário para Integração da Informação em Saúde**. [online] Disponível na Internet via WWW. URL: http://www.datasus.gov.br/prc/prcdown.htm. Arquivo capturado em 13 de julho de 2005.

OASIS Open - Organization for the Advancement of Structured Information Standards. **About UDDI.** [online] Disponível na Internet via WWW. URL: http://www.uddi.org/about.html. Arquivo consultado em 26 de março de 2005.

SINGH, Inderjeet; et al. **Designing Web Services with the J2EETM 1.4 Platform: JAX-RPC, SOAP, and XML Technologies.** California: Pearson Education, 2004.

SPERBERG-MCQUEEN, C. M. ; THOMPSON, H. **XML Schema Definition.** [online] Disponível na Internet via WWW. URL: http://www.w3.org/XML/Schema. Arquivo consultado em 15 de março de 2005.

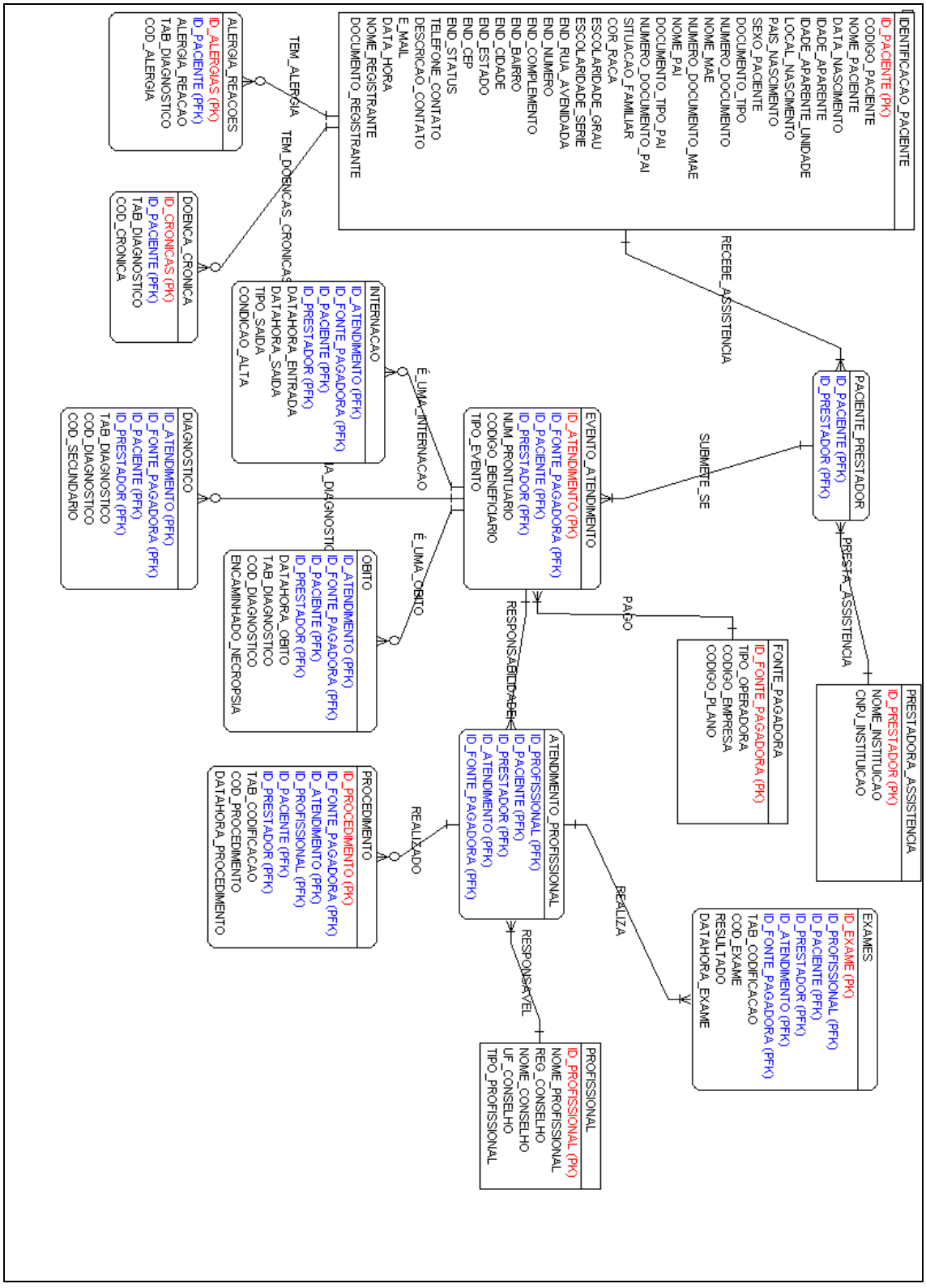
SUN - Sun Microsystems. **The Java Web Service Tutorial**. [online] Disponível na Internet via WWW. URL: http://java.sun.com/webservices/docs/1.5/tutorial/doc/JavaWSTutorial.pdf. Arquivo capturado em 7 de abril de 2005. Alterado em 19 de novembro de 2004.

W3C - World Wide Web Consortium. **Web Services Activity Statement**. [online] Disponível na Internet via WWW. URL: http://www.w3.org/2002/ws/Activity. Arquivo consultado em 11 de março de 2005.

WEERAWARANA, S. et al**. Web Services Platform Architecture: SOAP, WSDL, WS-Policy, WS-Addressing, WS-BPEL, WS-Reliable Messaging, and More.** U. S : Prentice Hall PTR, 2005.

APÊNDICE A

Modelo da Base de Dados do Serviço Web – ProntuarioWS:



APÊNDICE B

Esquemas XDS do Serviço Web ProntuarioWS:

Documento: EventoAtendimento.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2005 rel. 3 U (http://www.altova.com) by
Anderson (particular use) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
<xs:element name="ATENDIMENTO">
<xs:annotation>
<xs:documentation>Comment describing your root
element</xs:documentation>
</xs:annotation>
<xs:complexType>
<xs:sequence maxOccurs="unbounded">
<xs:element name="CODIGOS">
<xs:complexType>
<xs:sequence>
<xs:element name="ID_ATENDIMENTO" type="xs:unsignedLong"
minOccurs="0"/>
<xs:element name="ID_PROFISSIONAL" type="xs:unsignedLong"/>
<xs:element name="ID_PACIENTE" type="xs:unsignedLong"/>
<xs:element name="ID_PRESTADOR" type="xs:unsignedLong"/>
<xs:element name="ID_FONTE_PAGADORA" type="xs:unsignedLong"/>
<xs:element name="TIPO_EVENTO" type="xs:int"/>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="EXAME" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="ID_EXAME" type="xs:unsignedLong" minOccurs="0"/>
<xs:element name="TAB_CODIFICACAO">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="10"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="COD_EXAME">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="10"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="RESULTADO" minOccurs="0">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="8"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="DATAHORA_EXAME" type="xs:dateTime"/>
```

```
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="E_UMA" minOccurs="0">
<xs:complexType>
<xs:choice>
<xs:element name="INTERNACAO">
<xs:complexType>
<xs:sequence>
<xs:element name="DATAHORA_ENTRADA" type="xs:dateTime"/>
<xs:element name="DATAHORA_SAIDA" type="xs:dateTime"/>
<xs:element name="TIPO_SAIDA">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="50"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="CONDICAO_ALTA">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<1xs:maxLength value="10"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="DIAGNOSTICO">
<xs:complexType>
<xs:sequence>
<xs:element name="TAB_DIAGNOSTICO">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="10"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="COD_DIAGNOSTICO">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="10"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="COD_SECUNDARIO">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="10"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="OBITO">
```

```
<xs:complexType>
<xs:sequence>
<xs:element name="DATAHORA_OBITO" type="xs:dateTime"/>
<xs:element name="TAB_DIAGNOSTICO">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="10"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:choice>
</xs:complexType>
</xs:element>
<xs:element name="PROCEDIMENTO" minOccurs="0" maxOccurs="unbounded">
<xs:complexType>
<xs:sequence>
<xs:element name="ID_PROCEDIMENTO" type="xs:unsignedLong"
minOccurs="0"/>
<xs:element name="TAB_CODIFICACAO">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="10"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="COD_PROCEDIMENTO">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="10"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="DATAHORA_PROCEDIMENTO" type="xs:dateTime"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

Documento: Fonte_Pagadora.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2005 rel. 3 U (http://www.altova.com) by
Anderson (particular use) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
<xs:element name="FONTE">
<xs:annotation>
<xs:documentation>Comment describing your root
element</xs:documentation>
```

```
</xs:annotation>
<xs:complexType>
<xs:sequence maxOccurs="unbounded">
<xs:element name="ID_FONTE" type="xs:unsignedLong" minOccurs="0"/>
<xs:element name="TIPO_OPERADORA" type="xs:int" minOccurs="0"/>
<xs:element name="CODIGO_EMPRESA" type="xs:unsignedLong"
minOccurs="0"/>
<xs:element name="CODIGO_PLANO" type="xs:integer" minOccurs="0"/>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

Documento: Identificacao_Paciente.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2005 rel. 3 U (http://www.altova.com) by
Anderson (particular use) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
<xs:element name="PACIENTE">
<xs:annotation>
<xs:documentation>Root Elemento Paciente</xs:documentation>
</xs:annotation>
<xs:complexType>
<xs:sequence maxOccurs="unbounded">
<xs:element name="IDENTIFICACAO">
<xs:complexType>
<xs:sequence>
<xs:element name="ID_PACIENTE" type="xs:unsignedLong" minOccurs="0"/>
<xs:element name="CODIGO_PACIENTE">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="20"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="NOME_PACIENTE">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="50"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="DATA_NASCIMENTO" type="xs:date" minOccurs="0"/>
<xs:element name="IDADE_APARENTE" type="xs:int" minOccurs="0"/>
<xs:element name="IDADE_APARENTE_UNIDADE" minOccurs="0">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="5"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="SEXO_PACIENTE">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="1"/>
```

```
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="DOCUMENTO_TIPO" minOccurs="0">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="2"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="NUMERO_DOCUMENTO" minOccurs="0">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="40"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="NOME_MAE">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="50"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="DOCUMENTO_TIPO_MAE" minOccurs="0">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="2"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="NUMERO_DOCUMENTO_MAE" minOccurs="0">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="40"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="NOME_PAI" minOccurs="0">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="50"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="DOCUMENTO_TIPO_PAI" minOccurs="0">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="2"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="NUMERO_DOCUMENTO_PAI" minOccurs="0">
<xs:simpleType>
```

```
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="40"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="SITUACAO_FAMILIAR">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="33"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="COR_RACA">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="13"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="ESCOLARIDADE_GRAU">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="31"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="ESCOLARIDADE_SERIE">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="7"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="ENDERECO">
<xs:complexType>
<xs:sequence>
<xs:element name="LOCAL_NASCIMENTO">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="7"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="PAIS_NASCIMENTO" minOccurs="0">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="3"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="END_RUA_AVENIDA" minOccurs="0">
```

```
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="50"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="END_NUMERO" minOccurs="0">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="10"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="END_COMPLEMENTO" minOccurs="0">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="20"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="END_BAIRRO" minOccurs="0">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="30"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="END_CIDADE" minOccurs="0">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="50"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="END_ESTADO" minOccurs="0">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="2"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="END_CEP" minOccurs="0">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="9"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="END_STATUS" minOccurs="0">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="1"/>
</xs:restriction>
```

```
</xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="CONTATO">
<xs:complexType>
<xs:sequence>
<xs:element name="TELEFONE_CONTATO" minOccurs="0">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="12"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="DESCRICAO_CONTATO" minOccurs="0">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="250"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="E_MAIL" minOccurs="0">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="80"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="REGISTRO">
<xs:complexType>
<xs:sequence>
<xs:element name="DATA_HORA" type="xs:dateTime" minOccurs="0"/>
<xs:element name="NOME_REGISTRANTE" minOccurs="0">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="50"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="DOCUMENTO_REGISTRANTE" minOccurs="0">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="40"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="ALERGIA_REACOES">
<xs:complexType>
<xs:sequence minOccurs="0" maxOccurs="unbounded">
```

```
<xs:element name="ID_ALERGIAS" type="xs:unsignedLong"/>
<xs:element name="ID_PACIENTE" type="xs:unsignedLong"/>
<xs:element name="ALERGIA_REACAO">
<xs:simpleType>
<xs:restriction base="xs:string"/>
</xs:simpleType>
</xs:element>
<xs:element name="TAB_DIAGNOSTICO">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="10"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="COD_ALERGIA">
<xs:simpleType>
<xs:restriction base="xs:string"/>
</xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
<xs:element name="DOENCA_CRONICA">
<xs:complexType>
<xs:sequence minOccurs="0" maxOccurs="unbounded">
<xs:element name="ID_CRONICAS" type="xs:unsignedLong"/>
<xs:element name="ID_PACIENTE" type="xs:unsignedLong"/>
<xs:element name="TAB_DIAGNOSTICO">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="10"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="COD_CRONICA">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="10"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

Documento: Prestadora_Assistencia.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2005 rel. 3 U (http://www.altova.com) by
Anderson (particular use) -->
<xs:schema xmlns="http://localhost:8080/ProntuarioService/Schema"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://localhost:8080/ProntuarioService/Schema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
```

```
<xs:element name="PRESTADOR">
<xs:annotation>
<xs:documentation>Comment describing your root
element</xs:documentation>
</xs:annotation>
<xs:complexType>
<xs:sequence maxOccurs="unbounded">
<xs:element name="ID_PRESTADOR" type="xs:unsignedLong" minOccurs="0"/>
<xs:element name="NOME_INSTITUICAO">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="50"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="CNPJ_INSTITUICAO">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="18"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

Documento: Profissional.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSpy v2005 rel. 3 U (http://www.altova.com) by
Anderson (particular use) -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
<xs:element name="PROFISSIONAL">
<xs:annotation>
<xs:documentation>Comment describing your root
element</xs:documentation>
</xs:annotation>
<xs:complexType>
<xs:sequence maxOccurs="unbounded">
<xs:element name="ID_PROFISSIONAL" type="xs:unsignedLong"
minOccurs="0"/>
<xs:element name="NOME_PROFISSIONAL">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="50"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="REG_CONSELHO">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="20"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
```

```
<xs:element name="NOME_CONSELHO">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="50"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="UF_CONSELHO">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="2"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
<xs:element name="TIPO_PROFISSIONAL">
<xs:simpleType>
<xs:restriction base="xs:string">
<xs:minLength value="1"/>
<xs:maxLength value="14"/>
</xs:restriction>
</xs:simpleType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>
```

APÊNDICE C

Código Fonte do Serviço Web ProntuarioWS.

Classes de Interface do Serviço:

Classe: Interface ProntuarioWSSEI.

```
package org.anderson.prontuario;

/**
 * This is the service endpoint interface for the ProntuarioWSweb
service.
 * Created 08/09/2005 15:23:26
 * @author anderson
 */
public interface ProntuarioWSSEI extends java.rmi.Remote {
    /**
     * Web service operation
     */
    public String enviaDadosPaciente(long idPrestador,
java.lang.String docXML) throws java.rmi.RemoteException;

    /**
     * Web service operation
     */
    public String buscarDadosPaciente(long idPaciente) throws
java.rmi.RemoteException;

    /**
     * Web service operation
     */
    public String enviaDadosPrestadora(java.lang.String docXML) throws
java.rmi.RemoteException;

    /**
     * Web service operation
     */
    public String buscarDadosPrestadora(long idPrestadora) throws
java.rmi.RemoteException;

    /**
     * Web service operation
     */
    public String enviaDadosFontePagadora(java.lang.String docXML)
throws java.rmi.RemoteException;

    /**
     * Web service operation
     */
    public String buscarDadosFontePagadora(long idFontePagadora)
throws java.rmi.RemoteException;

    /**
     * Web service operation
```

```
     */
    public String enviaDadosProfissional(java.lang.String docXML)
throws java.rmi.RemoteException;

    /**
     * Web service operation
     */
    public String buscarDadosProfissional(long idProfissional) throws
java.rmi.RemoteException;

    /**
     * Web service operation
     */
    public String enviaAtendimento(java.lang.String docXML) throws
java.rmi.RemoteException;

    /**
     * Web service operation
     */
    public String buscarAtendimento(long idAtendimento) throws
java.rmi.RemoteException;

}
```

Classe: ProntuarioWSImpl Implementação da Interface ProntuarioWSSEI.

```
package org.anderson.prontuario;

import org.anderson.prontuario.paciente.InserirDadosPaciente;
import org.anderson.prontuario.paciente.BuscarDadosPaciente;
import org.anderson.prontuario.prestadora.InserirDadosPrestadora;
import org.anderson.prontuario.prestadora.BuscarDadosPrestadora;
import org.anderson.prontuario.fonte.InserirDadosFontePagadora;
import org.anderson.prontuario.fonte.BuscarDadosFontePagadora;
import org.anderson.prontuario.profissional.InserirDadosProfissional;
import org.anderson.prontuario.profissional.BuscarDadosProfissional;
import org.anderson.prontuario.atendimento.InserirDadosAtendimento;
import org.anderson.prontuario.atendimento.BuscarDadosAtendimento;


/**
 * This is the implementation bean class for the ProntuarioWS web
service.
 * Created 08/09/2005 15:23:26
 * @author anderson
 */

public class ProntuarioWSImpl implements ProntuarioWSSEI {
    private InserirDadosPaciente inserirPaciente;
    private BuscarDadosPaciente buscarPaciente;
    private InserirDadosPrestadora inserirPrestador;
    private BuscarDadosPrestadora buscarPrestador;
    private InserirDadosFontePagadora inserirFonte;
    private BuscarDadosFontePagadora buscarFonte;
    private InserirDadosProfissional inserirProfissional;
    private BuscarDadosProfissional buscarProfissional;
    private InserirDadosAtendimento inserirAtendimento;
    private BuscarDadosAtendimento buscarAtendimento;
```

```
    // Enter web service operations here. (Popup menu: Web Service-
>Add Operation)
    /**
     * Web service operation
     */
    public String enviaDadosPaciente(long idPrestador,
java.lang.String docXML) {
             // TODO implement operation
        inserirPaciente = new InserirDadosPaciente(docXML);

        return inserirPaciente.insereDados();

    }

    /**
     * Web service operation
     */
    public String buscarDadosPaciente(long idPaciente) {
        // TODO implement operation
        buscarPaciente = new BuscarDadosPaciente();

        return buscarPaciente.buscarPaciente(idPaciente);
    }

    /**
     * Web service operation
     */
    public String enviaDadosPrestadora(java.lang.String docXML) {
        // TODO implement operation

        inserirPrestador = new InserirDadosPrestadora(docXML);

        return inserirPrestador.insereDados();
    }

    /**
     * Web service operation
     */
    public String buscarDadosPrestadora(long idPrestadora) {
        // TODO implement operation
        buscarPrestador = new BuscarDadosPrestadora();

        return buscarPrestador.buscarPrestadora(idPrestadora);
    }

    /**
     * Web service operation
     */
    public String enviaDadosFontePagadora(java.lang.String docXML) {

        inserirFonte = new InserirDadosFontePagadora(docXML);

        return inserirFonte.insereDados();
    }

    /**
     * Web service operation
     */
    public String buscarDadosFontePagadora(long idFontePagadora) {

        buscarFonte = new BuscarDadosFontePagadora();
```

```
        return buscarFonte.buscarFontePagadora(idFontePagadora);
    }

    /**
     * Web service operation
     */
    public String enviaDadosProfissional(java.lang.String docXML) {
        // TODO implement operation
        inserirProfissional = new InserirDadosProfissional(docXML);

        return inserirProfissional.insereDados();
    }

    /**
     * Web service operation
     */
    public String buscarDadosProfissional(long idProfissional) {
        // TODO implement operation
        buscarProfissional = new BuscarDadosProfissional();

        return buscarProfissional.buscarProfissional(idProfissional);
    }

    /**
     * Web service operation
     */
    public String enviaAtendimento(java.lang.String docXML) {
        // TODO implement operation

        inserirAtendimento = new InserirDadosAtendimento(docXML);
        return inserirAtendimento.insereDados();
    }

    /**
     * Web service operation
     */
    public String buscarAtendimento(long idAtendimento) {
        // TODO implement operation
        buscarAtendimento  = new BuscarDadosAtendimento();
        return buscarAtendimento.buscarAtendimento(idAtendimento)  ;
    }


}
```

Classes que realizam as operações do Serviço:

Classe: InserirDadosPrestadora.

```
/*
 * inserirDadosPrestadora.java
 *
 * Created on 28 de Outubro de 2005, 22:20
 *
 * To change this template, choose Tools | Options and locate the
template under
```

```
 * the Source Creation and Management node. Right-click the template
and choose
 * Open. You can then make changes to the template in the Source
Editor.
 */

package org.anderson.prontuario.prestadora;

import java.io.*;
import javax.xml.transform.stream.*;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Unmarshaller;
import java.util.List;
import java.util.Date;
import java.util.Calendar;
import java.text.SimpleDateFormat;
import java.sql.ResultSet;
import org.anderson.prontuario.banco.Banco;
/**
 *
 * @author anderson
 */
public class InserirDadosPrestadora {
    private JAXBContext jc;
    private Unmarshaller u;
    private StringBuffer xmlStr;
    private String dados;
    private Banco BD;
    private String SQL="";
    private String resultado="0|";

    /** Creates a new instance of inserirDadosPrestadora */
    public InserirDadosPrestadora(String docXML) {

         try
         {
            jc =
JAXBContext.newInstance("org.anderson.prontuario.prestadora");
            u = jc.createUnmarshaller();
            u.setValidating(true);
            xmlStr = new StringBuffer(docXML);
            BD = new Banco();
         }
         catch(JAXBException je)
         {
                je.printStackTrace();

         }
         catch(Exception ex)
         {
             ex.printStackTrace();
         }

    }

    public String insereDados()
    {   int i;
        ResultSet rs;
        int idPrestdora;
```

```
        String result = BD.criarConexao();
        if( result != "0")
            return "1|Erro ao conectar-se a Base de Dados: ParÃ¢metros
de conexÃ£o Incorretos!";


        try
        {
             PRESTADOR prestador = (PRESTADOR) u.unmarshal(new
StreamSource(new StringReader(xmlStr.toString())));
             List prestadorList =
prestador.getIDPRESTADORAndNOMEINSTITUICAOAndCNPJINSTITUICAO();

             //Buscar ID para Paciente, Alergia e Cronicas
             SQL = "SELECT MAX(ID_PRESTADOR) FROM
PRESTADORA_ASSISTENCIA";

             rs = BD.executarSQL(SQL);

             if(rs != null)
             {

                 rs.next();
                 idPrestdora = rs.getInt(1) + 1;
             }
             else
                 idPrestdora = 1;

            for( i = 0; i+1 <prestadorList.size();i+=3 ) {

                //PRESTADOR.IDPRESTADOR idPres =
(PRESTADOR.IDPRESTADOR) prestadorList.get(i);
                PRESTADOR.NOMEINSTITUICAO nomeInst =
(PRESTADOR.NOMEINSTITUICAO) prestadorList.get(i+1);
                PRESTADOR.CNPJINSTITUICAO cnpj =
(PRESTADOR.CNPJINSTITUICAO) prestadorList.get(i+2);

                 SQL = "INSERT INTO PRESTADORA_ASSISTENCIA(
ID_PRESTADOR, NOME_INSTITUICAO, CNPJ_INSTITUICAO)" +
                         "VALUES(" + idPrestdora +
                         ",'" + nomeInst.getValue() +
                         "','" + cnpj.getValue() +
                         "')";

                 if( BD.executarUpdate(SQL) != 1)
                     return "1|Ocorreu um erro na inserÃ§Ã£o da
PRESTADORA: " +
                        nomeInst.getValue();

                 //Retorna cÃ³digo do Paciente indicando sucesso
                 resultado+= idPrestdora+"|";
                 idPrestdora++;

            }


        }
        catch(JAXBException je)
        {
           return "1|JAXBException je= " + je.getMessage();
        }
```

```
        catch(ClassCastException ex)
        {
           return "1|ClassCastException = " + ex.getClass() +"\n"+
ex.getCause() +"\n"+ ex.getMessage();
        }
        catch(Exception ex)
        {
            return "1|Outras exceÃ§Ãµes= " + ex.getMessage();
        }
        finally{
            BD.desconecta();
        }
        return resultado;


    }

}
```

Classe: BuscarDadosPrestadora.

```
/*
 * BuscarDadosPrestadora.java
 *
 * Created on 28 de Outubro de 2005, 20:54
 *
 * To change this template, choose Tools | Options and locate the
template under
 * the Source Creation and Management node. Right-click the template
and choose
 * Open. You can then make changes to the template in the Source
Editor.
 */

package org.anderson.prontuario.prestadora;
import javax.xml.transform.dom.DOMResult;
import javax.xml.transform.stream.*;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
import java.util.List;
import java.util.Calendar;
import java.math.BigInteger;
import java.sql.ResultSet;
import org.anderson.prontuario.banco.Banco;
import java.io.*;


/**
 *
 * @author anderson
 */
public class BuscarDadosPrestadora {

private Banco BD;
private String SQL;
private ResultSet rs;
```

```
private String xmlStr;

    /** Creates a new instance of BuscarDadosPrestadora */
    public BuscarDadosPrestadora() {
         BD = new Banco();
    }

    public String buscarPrestadora(long idPrestadora){
        try
        {

            JAXBContext jc =
JAXBContext.newInstance("org.anderson.prontuario.prestadora");
            ObjectFactory ofact = new ObjectFactory();
            PRESTADOR prestador = ofact.createPRESTADOR();

            BD.criarConexao();

            SQL = "SELECT * FROM PRESTADORA_ASSISTENCIA " +
                    " WHERE ID_PRESTADOR = " + idPrestadora;
            rs = BD.executarSQL(SQL);



           //List paList = pa.getIDENTIFICACAOAndENDERECOAndCONTATO();

           if(rs.next())
           {

                PRESTADOR.IDPRESTADOR idPres =
ofact.createPRESTADORTypeIDPRESTADOR();
                PRESTADOR.NOMEINSTITUICAO nomeInst =
ofact.createPRESTADORTypeNOMEINSTITUICAO();
                PRESTADOR.CNPJINSTITUICAO cnpj =
ofact.createPRESTADORTypeCNPJINSTITUICAO();


                idPres.setValue(new BigInteger(
rs.getString("ID_PRESTADOR") ) );
                nomeInst.setValue( rs.getString("NOME_INSTITUICAO") );
                cnpj.setValue( rs.getString("CNPJ_INSTITUICAO") );


prestador.getIDPRESTADORAndNOMEINSTITUICAOAndCNPJINSTITUICAO().add(0,
idPres);

prestador.getIDPRESTADORAndNOMEINSTITUICAOAndCNPJINSTITUICAO().add(1,
nomeInst);

prestador.getIDPRESTADORAndNOMEINSTITUICAOAndCNPJINSTITUICAO().add(2,
cnpj);


                Marshaller m = jc.createMarshaller();
                StringWriter swriter = new StringWriter();

                m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
Boolean.TRUE);
                m.marshal(prestador,swriter);

                xmlStr = swriter.toString();
```

```
           }else
               return "1|Prestadora de Assistencia nÃ£o encontrada";


        }
        catch(JAXBException ex)
        {
           return "1|JAXBException= " + ex.getMessage();
        }
        catch(Exception ex)
        {
           return "1|Exception= " + ex.getMessage();
        }

        return "0|" + xmlStr;

    }



}
```

Classe: InserirDadosProfissional.

```
/*
 * InserirDadosProfissional.java
 *
 * Created on 29 de Outubro de 2005, 12:25
 *
 * To change this template, choose Tools | Options and locate the
template under
 * the Source Creation and Management node. Right-click the template
and choose
 * Open. You can then make changes to the template in the Source
Editor.
 */

package org.anderson.prontuario.profissional;

import java.io.*;
import javax.xml.transform.stream.*;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Unmarshaller;
import java.util.List;
import java.util.Date;
import java.util.Calendar;
import java.text.SimpleDateFormat;
import java.sql.ResultSet;
import org.anderson.prontuario.banco.Banco;
/**
 *
 * @author anderson
 */
public class InserirDadosProfissional {
    private JAXBContext jc;
```

```
    private Unmarshaller u;
    private StringBuffer xmlStr;
    private String dados;
    private Banco BD;
    private String SQL="";
    private String resultado="0|";
    /** Creates a new instance of InserirDadosProfissional */

    public InserirDadosProfissional(String docXML) {
         try
         {
            jc =
JAXBContext.newInstance("org.anderson.prontuario.profissional");
            u = jc.createUnmarshaller();
            u.setValidating(true);
            xmlStr = new StringBuffer(docXML);
            BD = new Banco();
         }
         catch(JAXBException je)
         {
                je.printStackTrace();

         }
         catch(Exception ex)
         {
             ex.printStackTrace();
         }
    }

    public String insereDados()
    {   int i;
        ResultSet rs;
        int idProfissional;

        String result = BD.criarConexao();
        if( result != "0")
            return "1|Erro ao conectar-se a Base de Dados: ParÃ¢metros
de conexÃ£o Incorretos!";


        try
        {
             PROFISSIONAL profissional  = (PROFISSIONAL)
u.unmarshal(new StreamSource(new StringReader(xmlStr.toString())));
             List profissionalList =
profissional.getIDPROFISSIONALAndNOMEPROFISSIONALAndREGCONSELHO();

             //Buscar ID para PROFISSIONAL
             SQL = "SELECT MAX(ID_PROFISSIONAL) FROM PROFISSIONAL";

             rs = BD.executarSQL(SQL);

             if(rs != null)
             {
                 rs.next();
                 idProfissional = rs.getInt(1) + 1;
             }
             else
                 idProfissional = 1;

            for( i = 0; i+1 <profissionalList.size();i+=6 ) {
```

```
                //PROFISSIONAL.IDPROFISSIONAL idProf =  (
PROFISSIONAL.IDPROFISSIONAL ) profissionalList.get(i);
                PROFISSIONAL.NOMEPROFISSIONAL nomProf =  (
PROFISSIONAL.NOMEPROFISSIONAL ) profissionalList.get(i+1);
                PROFISSIONAL.REGCONSELHO numReg =  (
PROFISSIONAL.REGCONSELHO ) profissionalList.get(i+2);
                PROFISSIONAL.NOMECONSELHO nomeCons =  (
PROFISSIONAL.NOMECONSELHO ) profissionalList.get(i+3);
                PROFISSIONAL.UFCONSELHO ufCons =  (
PROFISSIONAL.UFCONSELHO ) profissionalList.get(i+4);
                PROFISSIONAL.TIPOPROFISSIONAL tipoProf =  (
PROFISSIONAL.TIPOPROFISSIONAL ) profissionalList.get(i+5);


                SQL = "INSERT INTO PROFISSIONAL( ID_PROFISSIONAL,
NOME_PROFISSIONAL, REG_CONSELHO, NOME_CONSELHO, UF_CONSELHO,
TIPO_PROFISSIONAL)" +
                        "VALUES(" + idProfissional +
                        " ,'" + nomProf.getValue() +
                        "','" + numReg.getValue() +
                        "','" + nomeCons.getValue() +
                        "','" + ufCons.getValue() +
                        "','" + tipoProf.getValue() +
                        "')";

                 if( BD.executarUpdate(SQL) != 1)
                     return "1|Ocorreu um erro na inserÃ§Ã£o do
PROFISSIONAL: " +
                        nomProf.getValue();

                 //Retorna código do Profissional indicando sucesso
                 resultado+= idProfissional+"|";
                 idProfissional++;

             }


        }
        catch(JAXBException je)
        {
           return "1|JAXBException je= " + je.getMessage();
        }
        catch(ClassCastException ex)
        {
           return "1|ClassCastException = " + ex.getClass() +"\n"+
ex.getCause() +"\n"+ ex.getMessage();
        }
        catch(Exception ex)
        {
            return "1|Outras exceÃ§Ãµes= " + ex.getMessage();
        }
        finally{
            BD.desconecta();
        }
        return resultado;


    }

}
```

Classe: BuscarDadosProfissional.

```
/*
 * BuscarDadosProfissional.java
 *
 * Created on 29 de Outubro de 2005, 12:26
 *
 * To change this template, choose Tools | Options and locate the
template under
 * the Source Creation and Management node. Right-click the template
and choose
 * Open. You can then make changes to the template in the Source
Editor.
 */

package org.anderson.prontuario.profissional;

import javax.xml.transform.dom.DOMResult;
import javax.xml.transform.stream.*;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
import java.util.List;
import java.util.Calendar;
import java.math.BigInteger;
import java.sql.ResultSet;
import org.anderson.prontuario.banco.Banco;
import java.io.*;
/**
 *
 * @author anderson
 */
public class BuscarDadosProfissional {
    private Banco BD;
    private String SQL;
    private ResultSet rs;
    private String xmlStr;

    /** Creates a new instance of BuscarDadosProfissional */
    public BuscarDadosProfissional() {
        BD = new Banco();
    }

    public String buscarProfissional(long idProfissional){
        try
        {

            JAXBContext jc =
JAXBContext.newInstance("org.anderson.prontuario.profissional");
            ObjectFactory ofact = new ObjectFactory();
            PROFISSIONAL profissional = ofact.createPROFISSIONAL();

            BD.criarConexao();

            SQL = "SELECT * FROM PROFISSIONAL " +
                    " WHERE ID_PROFISSIONAL = " + idProfissional;
            rs = BD.executarSQL(SQL);


           if(rs.next())
```

```
           {

                PROFISSIONAL.IDPROFISSIONAL idProf =
ofact.createPROFISSIONALTypeIDPROFISSIONAL();
                PROFISSIONAL.NOMEPROFISSIONAL nomProf =
ofact.createPROFISSIONALTypeNOMEPROFISSIONAL();
                PROFISSIONAL.REGCONSELHO numReg =
ofact.createPROFISSIONALTypeREGCONSELHO();
                PROFISSIONAL.NOMECONSELHO nomeCons =
ofact.createPROFISSIONALTypeNOMECONSELHO();
                PROFISSIONAL.UFCONSELHO ufCons =
ofact.createPROFISSIONALTypeUFCONSELHO();
                PROFISSIONAL.TIPOPROFISSIONAL tipoProf =
ofact.createPROFISSIONALTypeTIPOPROFISSIONAL();


                idProf.setValue(new BigInteger(
rs.getString("ID_PROFISSIONAL") ) );
                nomProf.setValue( rs.getString("NOME_PROFISSIONAL") );
                numReg.setValue( rs.getString("REG_CONSELHO") );
                nomeCons.setValue( rs.getString("NOME_CONSELHO") );
                ufCons.setValue( rs.getString("UF_CONSELHO") );
                tipoProf.setValue( rs.getString("TIPO_PROFISSIONAL")
);



profissional.getIDPROFISSIONALAndNOMEPROFISSIONALAndREGCONSELHO().add(
0, idProf);

profissional.getIDPROFISSIONALAndNOMEPROFISSIONALAndREGCONSELHO().add(
1, nomProf);

profissional.getIDPROFISSIONALAndNOMEPROFISSIONALAndREGCONSELHO().add(
2, numReg);

profissional.getIDPROFISSIONALAndNOMEPROFISSIONALAndREGCONSELHO().add(
3, nomeCons);

profissional.getIDPROFISSIONALAndNOMEPROFISSIONALAndREGCONSELHO().add(
4, ufCons);

profissional.getIDPROFISSIONALAndNOMEPROFISSIONALAndREGCONSELHO().add(
5, tipoProf);

                Marshaller m = jc.createMarshaller();
                StringWriter swriter = new StringWriter();

                m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
Boolean.TRUE);
                m.marshal(profissional,swriter);

                xmlStr = swriter.toString();


           }else
               return "1|Profissional nÃ£o encontrada";


        }
        catch(JAXBException ex)
```

```
        {
           return "1|JAXBException= " + ex.getMessage();
        }
        catch(Exception ex)
        {
           return "1|Exception= " + ex.getMessage();
        }

        return "0|" + xmlStr;

    }
}
```

Classe: InserirDadosFontePagadora.

```
/*
 * InserirDadosFontePagadora.java
 *
 * Created on 29 de Outubro de 2005, 10:27
 *
 * To change this template, choose Tools | Options and locate the
template under
 * the Source Creation and Management node. Right-click the template
and choose
 * Open. You can then make changes to the template in the Source
Editor.
 */

package org.anderson.prontuario.fonte;

import java.io.*;
import javax.xml.transform.stream.*;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Unmarshaller;
import java.util.List;
import java.util.Date;
import java.util.Calendar;
import java.text.SimpleDateFormat;
import java.sql.ResultSet;
import org.anderson.prontuario.banco.Banco;
/**
 *
 * @author anderson
 */
public class InserirDadosFontePagadora {

    private JAXBContext jc;
    private Unmarshaller u;
    private StringBuffer xmlStr;
    private String dados;
    private Banco BD;
    private String SQL="";
    private String resultado="0|";

    /** Creates a new instance of InserirDadosFontePagadora */
    public InserirDadosFontePagadora(String docXML) {
        try
         {
```

```
            jc =
JAXBContext.newInstance("org.anderson.prontuario.fonte");
            u = jc.createUnmarshaller();
            u.setValidating(true);
            xmlStr = new StringBuffer(docXML);
            BD = new Banco();
         }
         catch(JAXBException je)
         {
                je.printStackTrace();

         }
         catch(Exception ex)
         {
             ex.printStackTrace();
         }

    }

    public String insereDados()
    {   int i;
        ResultSet rs;
        int idFonte;

        String result = BD.criarConexao();
        if( result != "0")
            return "1|Erro ao conectar-se a Base de Dados: ParÃ¢metros
de conexÃ£o Incorretos!";


        try
        {
             FONTE fonte  = (FONTE) u.unmarshal(new StreamSource(new
StringReader(xmlStr.toString())));
             List fonteList =
fonte.getIDFONTEAndTIPOOPERADORAAndCODIGOEMPRESA();

             //Buscar ID para FONTE PAGADORA
             SQL = "SELECT MAX(ID_FONTE_PAGADORA) FROM
FONTE_PAGADORA";

             rs = BD.executarSQL(SQL);

             if(rs != null)
             {

                 rs.next();
                 idFonte = rs.getInt(1) + 1;
             }
             else
                 idFonte = 1;

            for( i = 0; i+1 <fonteList.size();i+=4 ) {

               //FONTE.IDFONTE idFon =  (FONTE.IDFONTE)
fonteList.get(i);
                FONTE.TIPOOPERADORA tipoOper =  (FONTE.TIPOOPERADORA )
fonteList.get(i+1);
                FONTE.CODIGOEMPRESA codEmp =  (FONTE.CODIGOEMPRESA)
fonteList.get(i+2);
```

```
                FONTE.CODIGOPLANO codPlan =  ( FONTE.CODIGOPLANO )
fonteList.get(i+3);


                SQL = "INSERT INTO FONTE_PAGADORA( ID_FONTE_PAGADORA,
TIPO_OPERADORA, CODIGO_EMPRESA, CODIGO_PLANO)" +
                         "VALUES(" + idFonte +
                         " ,'" + tipoOper.getValue() +
                         "','" + codEmp.getValue() +
                         "','" + codPlan.getValue() +
                         "')";

                 if( BD.executarUpdate(SQL) != 1)
                     return "1|Ocorreu um erro na inserÃ§Ã£o da FONTE
PAGADORA: " +
                        codEmp.getValue();

                 //Retorna cÃ³digo da Fonte Pagadora indicando sucesso
                 resultado+= idFonte+"|";
                 idFonte++;

            }


        }
        catch(JAXBException je)
        {
           return "1|JAXBException je= " + je.getMessage();
        }
        catch(ClassCastException ex)
        {
           return "1|ClassCastException = " + ex.getClass() +"\n"+
ex.getCause() +"\n"+ ex.getMessage();
        }
        catch(Exception ex)
        {
            return "1|Outras exceÃ§Ãµes= " + ex.getMessage();
        }
        finally{
            BD.desconecta();
        }
        return resultado;


    }

}
```

Classe: BuscarDadosFontePagadora.

```
/*
 * BuscarDadosFontePagadora.java
 *
 * Created on 29 de Outubro de 2005, 10:27
 *
 * To change this template, choose Tools | Options and locate the
template under
 * the Source Creation and Management node. Right-click the template
and choose
```

```
 * Open. You can then make changes to the template in the Source
Editor.
 */

package org.anderson.prontuario.fonte;

import javax.xml.transform.dom.DOMResult;
import javax.xml.transform.stream.*;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
import java.util.List;
import java.util.Calendar;
import java.math.BigInteger;
import java.sql.ResultSet;
import org.anderson.prontuario.banco.Banco;
import java.io.*;

/**
 *
 * @author anderson
 */
public class BuscarDadosFontePagadora {

    private Banco BD;
    private String SQL;
    private ResultSet rs;
    private String xmlStr;


    /** Creates a new instance of BuscarDadosFontePagadora */
    public BuscarDadosFontePagadora() {
         BD = new Banco();
    }

    public String buscarFontePagadora(long idFonte){
        try
        {

            JAXBContext jc =
JAXBContext.newInstance("org.anderson.prontuario.fonte");
            ObjectFactory ofact = new ObjectFactory();
            FONTE fonte = ofact.createFONTE();

            BD.criarConexao();

            SQL = "SELECT * FROM FONTE_PAGADORA " +
                    " WHERE ID_FONTE_PAGADORA = " + idFonte;
            rs = BD.executarSQL(SQL);




          if(rs.next())
          {

                FONTE.IDFONTE idFon =  ofact.createFONTETypeIDFONTE();
                FONTE.TIPOOPERADORA tipoOper =
ofact.createFONTETypeTIPOOPERADORA();
                FONTE.CODIGOEMPRESA codEmp =
ofact.createFONTETypeCODIGOEMPRESA();
```

```
                FONTE.CODIGOPLANO codPlan =
ofact.createFONTETypeCODIGOPLANO();

                idFon.setValue(new BigInteger(
rs.getString("ID_FONTE_PAGADORA") ) );
                tipoOper.setValue( rs.getInt("TIPO_OPERADORA") );
                codEmp.setValue(new BigInteger(
rs.getString("CODIGO_EMPRESA") ) );
                codPlan.setValue(new BigInteger(
rs.getString("CODIGO_PLANO") ) );


fonte.getIDFONTEAndTIPOOPERADORAAndCODIGOEMPRESA().add(0, idFon);

fonte.getIDFONTEAndTIPOOPERADORAAndCODIGOEMPRESA().add(1, tipoOper);

fonte.getIDFONTEAndTIPOOPERADORAAndCODIGOEMPRESA().add(2, codEmp);

fonte.getIDFONTEAndTIPOOPERADORAAndCODIGOEMPRESA().add(3, codPlan);


                Marshaller m = jc.createMarshaller();
                StringWriter swriter = new StringWriter();

                m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
Boolean.TRUE);
                m.marshal(fonte,swriter);

                xmlStr = swriter.toString();


           }else
               return "1|Fonte Pagadora nÃ£o encontrada";


        }
        catch(JAXBException ex)
        {
           return "1|JAXBException= " + ex.getMessage();
        }
        catch(Exception ex)
        {
           return "1|Exception= " + ex.getMessage();
        }

        return "0|" + xmlStr;

    }

}
```

Classe: InserirDadosPaciente.

```
/*
 * InserirDadosPaciente lerDadosPaciente.java
 *
 * Created on 8 de Setembro de 2005, 22:14
 *
 * To change this template, choose Tools | Options and locate the
template under
```

```
 * the Source Creation and Management node. Right-click the template
and choose
 * Open. You can then make changes to the template in the Source
Editor.
 */

package org.anderson.prontuario.paciente;

import java.io.*;
import javax.xml.transform.stream.*;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Unmarshaller;
import java.util.List;
import java.util.Date;
import java.util.Calendar;
import java.text.SimpleDateFormat;
import java.sql.ResultSet;
import org.anderson.prontuario.banco.Banco;




/**
 *
 * @author anderson
 */
public class InserirDadosPaciente {
    private JAXBContext jc;
    private Unmarshaller u;
    private PACIENTE pa;
    private PACIENTEType.IDENTIFICACAOType pacIdent;
    private StringBuffer xmlStr;
    private String dados;
    private Banco BD;
    private String SQL="";
    private String resultado="0|";



    /** Creates a new instance of lerDadosPaciente */
    public InserirDadosPaciente(String docXML) {

     try
     {
        jc =
JAXBContext.newInstance("org.anderson.prontuario.paciente");
        u = jc.createUnmarshaller();
        u.setValidating(true);
        xmlStr = new StringBuffer(docXML);
        BD = new Banco();
     }
     catch(JAXBException je)
     {
            je.printStackTrace();

     }
     catch(Exception ex)
     {
         ex.printStackTrace();
     }
```

```
    }


    public String insereDados()
    {   int i;
        ResultSet rs;
        int idPaciente, idAlergia=1, idCronica=1;
        Date data = new Date();
        Date hora = new Date();
        Calendar cDataNascimento;
        Calendar cDataHora;
        SimpleDateFormat formato = new SimpleDateFormat("yyyymmdd");
        SimpleDateFormat formatoTimeStamp = new
SimpleDateFormat("hhmmss");
        String dataNascimento="";
        String dataHora;

        String result = BD.criarConexao();
        if( result != "0")
            return "1|Erro ao conectar-se a Base de Dados: ParÃ¢metros
de conexÃ£o Incorretos!";


        try
        {
             pa = (PACIENTE) u.unmarshal(new StreamSource(new
StringReader(xmlStr.toString())));
             List identificacaoList =
pa.getIDENTIFICACAOAndENDERECOAndCONTATO();


             //Buscar ID para Paciente, Alergia e Cronicas
             SQL = "SELECT MAX(ID_PACIENTE) FROM
IDENTIFICACAO_PACIENTE";

             rs = BD.executarSQL(SQL);

             if(rs != null)
             {

                 rs.next();
                 idPaciente = rs.getInt(1) + 1;
             }
             else
                 idPaciente = 1;

            for( i = 0; i+1 < identificacaoList.size();i+=6 ) {

                PACIENTE.IDENTIFICACAOType identificacao =
(PACIENTE.IDENTIFICACAOType) identificacaoList.get(i);
                PACIENTE.ENDERECOType endereco =
(PACIENTE.ENDERECOType) identificacaoList.get(i+1);
                PACIENTE.CONTATOType contato = (PACIENTE.CONTATOType)
identificacaoList.get(i+2);
                PACIENTE.REGISTROType registro =
(PACIENTE.REGISTROType) identificacaoList.get(i+3);

                if (identificacao.getCODIGOPACIENTE() == null )
                    identificacao.setCODIGOPACIENTE("");
```

```
                if (identificacao.getNOMEPACIENTE() == null)
                    identificacao.setNOMEPACIENTE("");
                if( identificacao.getIDADEAPARENTEUNIDADE() == null)
                    identificacao.setIDADEAPARENTEUNIDADE("");
                if( endereco.getLOCALNASCIMENTO() == null)
                    endereco.setLOCALNASCIMENTO("");
                if( endereco.getPAISNASCIMENTO() == null)
                    endereco.setPAISNASCIMENTO("");
                if( identificacao.getSEXOPACIENTE() == null)
                    identificacao.setSEXOPACIENTE("");
                if( identificacao.getDOCUMENTOTIPO() == null)
                    identificacao.setDOCUMENTOTIPO("");
                if( identificacao.getNUMERODOCUMENTO() == null)
                    identificacao.setNUMERODOCUMENTO("");
                if( identificacao.getNOMEMAE() == null)
                    identificacao.setNOMEMAE("");
                if( identificacao.getDOCUMENTOTIPOMAE() == null)
                    identificacao.setDOCUMENTOTIPOMAE("");
                if( identificacao.getNUMERODOCUMENTOMAE() == null)
                    identificacao.setNUMERODOCUMENTOMAE("");
                if( identificacao.getNOMEPAI() == null)
                     identificacao.setNOMEPAI("");
                if( identificacao.getDOCUMENTOTIPOPAI() == null)
                    identificacao.setDOCUMENTOTIPOPAI("");
                if( identificacao.getNUMERODOCUMENTOPAI() == null)
                    identificacao.setNUMERODOCUMENTOPAI("");
                if( identificacao.getSITUACAOFAMILIAR() == null)
                    identificacao.setSITUACAOFAMILIAR("");
                if( identificacao.getCORRACA() == null)
                    identificacao.setCORRACA("");
                if( identificacao.getESCOLARIDADEGRAU() == null)
                    identificacao.setESCOLARIDADEGRAU("");
                if( identificacao.getESCOLARIDADESERIE() == null)
                    identificacao.setESCOLARIDADESERIE("");
                if( endereco.getENDRUAAVENIDA() == null)
                    endereco.setENDRUAAVENIDA("");
                if( endereco.getENDNUMERO() == null )
                    endereco.setENDNUMERO("");
                if( endereco.getENDCOMPLEMENTO() == null )
                    endereco.setENDCOMPLEMENTO("");
                if( endereco.getENDBAIRRO() == null)
                    endereco.setENDBAIRRO("");
                if( endereco.getENDCIDADE() == null )
                    endereco.setENDCIDADE("");
                if( endereco.getENDESTADO() == null )
                    endereco.setENDESTADO("");
                if( endereco.getENDCEP() == null )
                    endereco.setENDCEP("");
                if( endereco.getENDSTATUS() == null )
                    endereco.setENDSTATUS("");
                if( contato.getTELEFONECONTATO() == null )
                    contato.setTELEFONECONTATO("");
                if( contato.getDESCRICAOCONTATO()  == null)
                    contato.setDESCRICAOCONTATO("");
                if( contato.getEMAIL() == null )
                    contato.setEMAIL("");
                if(registro.getNOMEREGISTRANTE() == null)
                   registro.setNOMEREGISTRANTE("");
                if(registro.getDOCUMENTOREGISTRANTE() == null)
                   registro.setDOCUMENTOREGISTRANTE("");
```

```
                if( identificacao.getDATANASCIMENTO() == null )
                    dataNascimento = "null";
                else
                {
                    cDataNascimento =
identificacao.getDATANASCIMENTO();
                    int mes = cDataNascimento.MONTH + 1;
                    dataNascimento = cDataNascimento.YEAR + "-" + mes
+ "-" + cDataNascimento.DAY_OF_MONTH;
                    dataNascimento = "'"+dataNascimento+"'";

                }

                if( registro.getDATAHORA() == null )
                    dataHora = "null";
                else
                {

                    cDataHora = registro.getDATAHORA();
                    int mes2 = cDataHora.MONTH + 1;
                    data.setYear(cDataHora.YEAR);
                    data.setDate(cDataHora.DAY_OF_MONTH);
                    data.setMonth(mes2);

                    hora.setHours(cDataHora.HOUR_OF_DAY);
                    hora.setMinutes(cDataHora.MINUTE);
                    hora.setSeconds(cDataHora.SECOND);

                    dataHora = formato.format(data);
                    dataHora += formatoTimeStamp.format(data);
                    dataHora = "'"+dataHora+"'";

                }

                 SQL = "INSERT INTO IDENTIFICACAO_PACIENTE(" +
                         "ID_PACIENTE," +  "CODIGO_PACIENTE," +
"NOME_PACIENTE," + "DATA_NASCIMENTO," +  "IDADE_APARENTE," +
                         "IDADE_APARENTE_UNIDADE," +
"LOCAL_NASCIMENTO," + "PAIS_NASCIMENTO," + "SEXO_PACIENTE," +
"DOCUMENTO_TIPO," +
                         "NUMERO_DOCUMENTO," +  "NOME_MAE," +
"DOCUMENTO_TIPO_MAE," +  "NUMERO_DOCUMENTO_MAE," + "NOME_PAI," +
                         "DOCUMENTO_TIPO_PAI," +
"NUMERO_DOCUMENTO_PAI," + "SITUACAO_FAMILIAR," + "COR_RACA," +
"ESCOLARIDADE_GRAU," +
                         "ESCOLARIDADE_SERIE," +  "END_RUA_AVENIDA," +
"END_NUMERO," + "END_COMPLEMENTO," +  "END_BAIRRO," +
                         "END_CIDADE," +    "END_ESTADO," +
"END_CEP," +  "END_STATUS," + "TELEFONE_CONTATO," +
                         "DESCRICAO_CONTATO," + "E_MAIL," +
"DATA_HORA," + "NOME_REGISTRANTE," +  "DOCUMENTO_REGISTRANTE" +
                         ")VALUES(" +  idPaciente +
                         " ,'" + identificacao.getCODIGOPACIENTE() +
                         "','" + identificacao.getNOMEPACIENTE() +
                         "', "  + dataNascimento +
                         " , " + identificacao.getIDADEAPARENTE() +
                         " ,'" +
identificacao.getIDADEAPARENTEUNIDADE() +
                         "','" + endereco.getLOCALNASCIMENTO() +
```

```
                          "','" + endereco.getPAISNASCIMENTO() +
                          "','" + identificacao.getSEXOPACIENTE() +
                          "','" + identificacao.getDOCUMENTOTIPO() +
                          "','" + identificacao.getNUMERODOCUMENTO() +
                          "','" + identificacao.getNOMEMAE() +
                          "','" + identificacao.getDOCUMENTOTIPOMAE() +
                          "','" + identificacao.getNUMERODOCUMENTOMAE()
+
                          "','" + identificacao.getNOMEPAI() +
                          "','" + identificacao.getDOCUMENTOTIPOPAI() +
                          "','" + identificacao.getNUMERODOCUMENTOPAI()
+
                          "','" + identificacao.getSITUACAOFAMILIAR() +
                          "','" + identificacao.getCORRACA() +
                          "','" + identificacao.getESCOLARIDADEGRAU() +
                          "','" + identificacao.getESCOLARIDADESERIE()
+
                          "','" + endereco.getENDRUAAVENIDA() +
                          "','" + endereco.getENDNUMERO() +
                          "','" + endereco.getENDCOMPLEMENTO() +
                          "','" + endereco.getENDBAIRRO() +
                          "','" + endereco.getENDCIDADE() +
                          "','" + endereco.getENDESTADO() +
                          "','" + endereco.getENDCEP() +
                          "','" + endereco.getENDSTATUS() +
                          "','" + contato.getTELEFONECONTATO() +
                          "','" + contato.getDESCRICAOCONTATO() +
                          "','" + contato.getEMAIL() +
                          "', " + dataHora +
                          " ,'" + registro.getNOMEREGISTRANTE() +
                          "','" + registro.getDOCUMENTOREGISTRANTE() +
                          "')";


                    if( BD.executarUpdate(SQL) != 1)
                        return "1|Ocorreu um erro na inserÃ§Ã£o do
Paciente: " +
                                identificacao.getCODIGOPACIENTE() +"-
"+
                                identificacao.getNOMEPACIENTE();

                    PACIENTE.ALERGIAREACOESType alergias =
(PACIENTE.ALERGIAREACOESType) identificacaoList.get(i+4);
                    List alergiaList =
alergias.getIDALERGIASAndIDPACIENTEAndALERGIAREACAO();

                    for( int j = 0; j < alergiaList.size();j+=5 )
                    {

                         //PACIENTE.ALERGIAREACOESType.IDALERGIAS
idAlergia = (PACIENTE.ALERGIAREACOESType.IDALERGIAS)
alergiaList.get(j);
                         //PACIENTE.ALERGIAREACOESType.IDPACIENTE
idPaciente = (PACIENTE.ALERGIAREACOESType.IDPACIENTE)
alergiaList.get(j+1);
                         PACIENTE.ALERGIAREACOESType.ALERGIAREACAO rea
= (PACIENTE.ALERGIAREACOESType.ALERGIAREACAO) alergiaList.get(j+2);
                         PACIENTE.ALERGIAREACOESType.TABDIAGNOSTICO
tab = (PACIENTE.ALERGIAREACOESType.TABDIAGNOSTICO)
alergiaList.get(j+3);
```

```
                        PACIENTE.ALERGIAREACOESType.CODALERGIA
codAler = (PACIENTE.ALERGIAREACOESType.CODALERGIA)
alergiaList.get(j+4);

                        if (rea.getValue() == null)
                           rea.setValue("");

                        if( tab.getValue() == null)
                            tab.setValue("");

                        if( codAler.getValue() == null)
                            codAler.setValue("");


                        SQL = "INSERT INTO ALERGIA_REACOES(
ID_ALERGIAS, ID_PACIENTE, ALERGIA_REACAO, TAB_DIAGNOSTICO, COD_ALERGIA
)" +
                                "VALUES(" + idAlergia +
                                "," + idPaciente +
                                ",'" + rea.getValue() +
                                "','" + tab.getValue() +
                                "','" + codAler.getValue() +
                                "')";

                         if( BD.executarUpdate(SQL) != 1)
                             return "1|Ocorreu um erro na inserÃ§Ã£o
da Alergia a ReaÃ§Ãµes do Paciente: " +
                                   identificacao.getCODIGOPACIENTE() +"-
"+
                                   identificacao.getNOMEPACIENTE();

                         idAlergia++;

                    }

                    PACIENTE.DOENCACRONICAType cronicas =
(PACIENTE.DOENCACRONICAType) identificacaoList.get(i+5);
                    List cronicaList =
cronicas.getIDCRONICASAndIDPACIENTEAndTABDIAGNOSTICO();

                    for( int k = 0; k < cronicaList.size();k+=5 )
                    {

                        //PACIENTE.DOENCACRONICAType.IDCRONICAS
idCronica = ( PACIENTE.DOENCACRONICAType.IDCRONICAS)
cronicaList.get(k);
                        //PACIENTE.DOENCACRONICAType.IDPACIENTE
idPaciente = ( PACIENTE.DOENCACRONICAType.IDPACIENTE )
cronicaList.get(k+1);
                        PACIENTE.DOENCACRONICAType.TABDIAGNOSTICO
tabCronica = ( PACIENTE.DOENCACRONICAType.TABDIAGNOSTICO )
cronicaList.get(k+2);
                        PACIENTE.DOENCACRONICAType.CODCRONICA
codCronica = ( PACIENTE.DOENCACRONICAType.CODCRONICA )
cronicaList.get(k+3);

                        if( tabCronica.getValue() == null)
                            tabCronica.setValue("");
                        if( codCronica.getValue() == null)
                            codCronica.setValue("");
```

```
                            SQL = "INSERT INTO DOENCA_CRONICA(
ID_CRONICAS, ID_PACIENTE, TAB_DIAGNOSTICO, COD_CRONICA )" +
                                    "VALUES(" + idCronica +
                                    "," + idPaciente +
                                    ",'" + tabCronica.getValue() +
                                    "','" + codCronica.getValue() +
                                    "')";

                               if( BD.executarUpdate(SQL) != 1)
                                    return "1|Ocorreu um erro na inserÃ§Ã£o
da DoenÃ§a Cronica do Paciente: " +
                                      identificacao.getCODIGOPACIENTE() +"-
"+
                                      identificacao.getNOMEPACIENTE();

                                idCronica++;
                        }



                        idAlergia = 1;
                        idCronica = 1;

                        //Retorna cÃ³digo do Paciente indicando sucesso
                        resultado+= idPaciente +"|";

                        idPaciente++;


                    }
                }
                catch(JAXBException je)
                {
                   return "1|JAXBException je= " + je.getMessage();
                }
                catch(ClassCastException ex)
                {
                   return "1|ClassCastException = " + ex.getClass() +"\n"+
ex.getCause() +"\n"+ ex.getMessage();
                }
                catch(Exception ex)
                {
                    return "1|Outras exceÃ§Ãµes= " + ex.getMessage();
                }
                finally{
                    BD.desconecta();
                }
                return resultado;

            }

}
```

Classe: BuscarDadosPaciente.

```
/*
 * BuscarDadosPaciente.java
 *
 * Created on 12 de Outubro de 2005, 09:16
 *
```

```
 * To change this template, choose Tools | Options and locate the
template under
 * the Source Creation and Management node. Right-click the template
and choose
 * Open. You can then make changes to the template in the Source
Editor.
 */

package org.anderson.prontuario.paciente;

import javax.xml.transform.dom.DOMResult;
import javax.xml.transform.stream.*;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
import java.util.List;
import java.util.Calendar;
import java.math.BigInteger;
import java.sql.ResultSet;
import org.anderson.prontuario.banco.Banco;
import org.anderson.prontuario.paciente.*;
import java.io.*;

public class BuscarDadosPaciente {
private Banco BD;
private JAXBContext jc;
private String SQL;
private ResultSet rs;
private PACIENTE pa;
private ObjectFactory objFact;
private String xmlStr;
private PrintWriter saida;




    /** Creates a new instance of BuscarDadosPaciente */
    public BuscarDadosPaciente() {
        BD = new Banco();


        //u = jc.createMarshaller();
       // u..setValidating(true);
       // xmlStr = new StringBuffer(docXML);
    }

    public String buscarPaciente(long idPaciente)
    {
        //Calendar data = new Calendar.getInstance();
        try
        {
            jc =
JAXBContext.newInstance("org.anderson.prontuario.paciente");
            objFact = new ObjectFactory();
            pa = objFact.createPACIENTE();


            BD.criarConexao();

            SQL = "SELECT * FROM IDENTIFICACAO_PACIENTE " +
```

```
                    " WHERE ID_PACIENTE = " + idPaciente;
            rs = BD.executarSQL(SQL);


           //List paList = pa.getIDENTIFICACAOAndENDERECOAndCONTATO();

           PACIENTE.IDENTIFICACAO identificacao =
objFact.createPACIENTETypeIDENTIFICACAO();
           PACIENTE.ENDERECO endereco =
objFact.createPACIENTETypeENDERECO();
           PACIENTE.CONTATO contato =
objFact.createPACIENTETypeCONTATO();
           PACIENTE.REGISTRO registro =
objFact.createPACIENTETypeREGISTRO();

           if(rs.next())
           {
               identificacao.setIDPACIENTE( new BigInteger(
rs.getString("ID_PACIENTE") ) );
               if ( rs.getString("CODIGO_PACIENTE").compareTo("") !=
0)
                    identificacao.setCODIGOPACIENTE(
rs.getString("CODIGO_PACIENTE") );
               if ( rs.getString("NOME_PACIENTE").compareTo("") != 0)
                    identificacao.setNOMEPACIENTE(
rs.getString("NOME_PACIENTE") );
               identificacao.setDATANASCIMENTO( Calendar.getInstance()
);
               identificacao.setIDADEAPARENTE(
rs.getInt("IDADE_APARENTE") );
               if (
rs.getString("IDADE_APARENTE_UNIDADE").compareTo("") != 0)
                    identificacao.setIDADEAPARENTEUNIDADE(
rs.getString("IDADE_APARENTE_UNIDADE") );
               if ( rs.getString( "LOCAL_NASCIMENTO" ).compareTo("")
!= 0)
                    endereco.setLOCALNASCIMENTO( rs.getString(
"LOCAL_NASCIMENTO" ) );
               if ( rs.getString( "PAIS_NASCIMENTO" ).compareTo("") !=
0)
                    endereco.setPAISNASCIMENTO( rs.getString(
"PAIS_NASCIMENTO" ) );
               if ( rs.getString( "SEXO_PACIENTE" ).compareTo("") !=
0)
                    identificacao.setSEXOPACIENTE( rs.getString(
"SEXO_PACIENTE" ) );
               if ( rs.getString( "DOCUMENTO_TIPO" ) .compareTo("") !=
0)
                    identificacao.setDOCUMENTOTIPO( rs.getString(
"DOCUMENTO_TIPO" ) );
               if ( rs.getString( "NUMERO_DOCUMENTO" ).compareTo("")
!= 0)
                    identificacao.setNUMERODOCUMENTO( rs.getString(
"NUMERO_DOCUMENTO" ) );
               if ( rs.getString( "NOME_MAE" ).compareTo("") != 0)
                    identificacao.setNOMEMAE( rs.getString( "NOME_MAE"
) );
               if ( rs.getString( "DOCUMENTO_TIPO_MAE" ).compareTo("")
!= 0)
                    identificacao.setDOCUMENTOTIPOMAE( rs.getString(
"DOCUMENTO_TIPO_MAE" ) );
```

```
                if ( rs.getString( "NUMERO_DOCUMENTO_MAE"
).compareTo("") != 0)
                     identificacao.setNUMERODOCUMENTOMAE( rs.getString(
"NUMERO_DOCUMENTO_MAE" ) );
                if ( rs.getString( "NOME_PAI" ).compareTo("") != 0)
                     identificacao.setNOMEPAI( rs.getString( "NOME_PAI"
) );
                if ( rs.getString( "DOCUMENTO_TIPO_PAI" ).compareTo("")
!= 0)
                     identificacao.setDOCUMENTOTIPOPAI( rs.getString(
"DOCUMENTO_TIPO_PAI" ) );
                if ( rs.getString( "NUMERO_DOCUMENTO_PAI"
).compareTo("") != 0)
                     identificacao.setNUMERODOCUMENTOPAI( rs.getString(
"NUMERO_DOCUMENTO_PAI" )  );
                if ( rs.getString( "SITUACAO_FAMILIAR" ).compareTo("")
!= 0)
                     identificacao.setSITUACAOFAMILIAR( rs.getString(
"SITUACAO_FAMILIAR" )  );
                if ( rs.getString( "COR_RACA" ).compareTo("") != 0)
                     identificacao.setCORRACA( rs.getString( "COR_RACA"
)  );
                if ( rs.getString( "ESCOLARIDADE_GRAU" ).compareTo("")
!= 0)
                     identificacao.setESCOLARIDADEGRAU( rs.getString(
"ESCOLARIDADE_GRAU" ) );
                if ( rs.getString( "ESCOLARIDADE_SERIE" ).compareTo("")
!= 0)
                     identificacao.setESCOLARIDADESERIE( rs.getString(
"ESCOLARIDADE_SERIE" ) );
                if ( rs.getString( "END_RUA_AVENIDA" ).compareTo("") !=
0)
                     endereco.setENDRUAAVENIDA( rs.getString(
"END_RUA_AVENIDA" ) );
                if ( rs.getString( "END_NUMERO" ).compareTo("") != 0)
                     endereco.setENDNUMERO( rs.getString( "END_NUMERO"
) );
                if ( rs.getString( "END_COMPLEMENTO" ).compareTo("") !=
0)
                     endereco.setENDCOMPLEMENTO( rs.getString(
"END_COMPLEMENTO" ) );
                if ( rs.getString( "END_BAIRRO" ).compareTo("") != 0)
                     endereco.setENDBAIRRO( rs.getString( "END_BAIRRO"
) );
                if ( rs.getString( "END_CIDADE" ).compareTo("") != 0)
                     endereco.setENDCIDADE( rs.getString( "END_CIDADE"
) );
                if ( rs.getString( "END_ESTADO" ).compareTo("") != 0)
                     endereco.setENDESTADO( rs.getString( "END_ESTADO"
) );
                if ( rs.getString("END_CEP").compareTo("") != 0)
                     endereco.setENDCEP( rs.getString( "END_CEP" ) );
                if ( rs.getString("END_STATUS").compareTo("") != 0)
                     endereco.setENDSTATUS( rs.getString( "END_STATUS"
) );
                if ( rs.getString("TELEFONE_CONTATO").compareTo("") !=
0)
                     contato.setTELEFONECONTATO( rs.getString(
"TELEFONE_CONTATO" ) );
                if ( rs.getString("DESCRICAO_CONTATO").compareTo("") !=
0)
```

```
                    contato.setDESCRICAOCONTATO( rs.getString(
"DESCRICAO_CONTATO" ) );
               if ( rs.getString("E_MAIL").compareTo("") != 0)
                    contato.setEMAIL( rs.getString( "E_MAIL" ) );
               registro.setDATAHORA(Calendar.getInstance());
               if ( rs.getString("NOME_REGISTRANTE").compareTo("") !=
0)
                    registro.setNOMEREGISTRANTE(  rs.getString(
"NOME_REGISTRANTE" ) );
               if (
rs.getString("DOCUMENTO_REGISTRANTE").compareTo("")!= 0)
                    registro.setDOCUMENTOREGISTRANTE( rs.getString(
"DOCUMENTO_REGISTRANTE" ) );

               pa.getIDENTIFICACAOAndENDERECOAndCONTATO().add(0,
identificacao);
               pa.getIDENTIFICACAOAndENDERECOAndCONTATO().add(1,
endereco);
               pa.getIDENTIFICACAOAndENDERECOAndCONTATO().add(2,
contato);
               pa.getIDENTIFICACAOAndENDERECOAndCONTATO().add(3,
registro);

               /*paList.add(identificacao);
               paList.add(endereco);
               paList.add(contato);
               paList.add(registro);
               */

                SQL = "SELECT * FROM ALERGIA_REACOES " +
                    " WHERE ID_PACIENTE = " + idPaciente;
                rs = BD.executarSQL(SQL);


                PACIENTE.ALERGIAREACOES alergias =
objFact.createPACIENTETypeALERGIAREACOES();

                List alergiasList =
alergias.getIDALERGIASAndIDPACIENTEAndALERGIAREACAO();


                while(rs.next())
                {
                    PACIENTE.ALERGIAREACOES.IDPACIENTE idPac =
objFact.createPACIENTETypeALERGIAREACOESTypeIDPACIENTE();
                    PACIENTE.ALERGIAREACOES.IDALERGIAS idAler =
objFact.createPACIENTETypeALERGIAREACOESTypeIDALERGIAS();
                    PACIENTE.ALERGIAREACOES.ALERGIAREACAO rea =
objFact.createPACIENTETypeALERGIAREACOESTypeALERGIAREACAO();
                    PACIENTE.ALERGIAREACOES.TABDIAGNOSTICO tab =
objFact.createPACIENTETypeALERGIAREACOESTypeTABDIAGNOSTICO();
                    PACIENTE.ALERGIAREACOES.CODALERGIA codAler =
objFact.createPACIENTETypeALERGIAREACOESTypeCODALERGIA();


                    idPac.setValue( new BigInteger(
rs.getString("ID_PACIENTE") ) );
                    idAler.setValue( new BigInteger(
rs.getString("ID_ALERGIAS") ) );
                    rea.setValue( rs.getString("ALERGIA_REACAO") );
                    tab.setValue( rs.getString("TAB_DIAGNOSTICO") );
```

```
                    codAler.setValue( rs.getString("COD_ALERGIA") );

                    alergiasList.add(idAler);
                    alergiasList.add(idPac);
                    alergiasList.add(rea);
                    alergiasList.add(tab);
                    alergiasList.add(codAler);

                }

                pa.getIDENTIFICACAOAndENDERECOAndCONTATO().add(4 ,
alergiasList);
                //paList.add(alergiasList);


                PACIENTE.DOENCACRONICA cronicas =
objFact.createPACIENTETypeDOENCACRONICA();

                List cronicaList =
cronicas.getIDCRONICASAndIDPACIENTEAndTABDIAGNOSTICO();


                while(rs.next())
                {


                    PACIENTE.DOENCACRONICA.IDCRONICAS idPac  =
objFact.createPACIENTETypeDOENCACRONICATypeIDCRONICAS();
                    PACIENTE.DOENCACRONICA.IDPACIENTE idCron =
objFact.createPACIENTETypeDOENCACRONICATypeIDPACIENTE();
                    PACIENTE.DOENCACRONICA.TABDIAGNOSTICO tabCron =
objFact.createPACIENTETypeDOENCACRONICATypeTABDIAGNOSTICO();
                    PACIENTE.DOENCACRONICAType.CODCRONICA codCronica =
objFact.createPACIENTETypeDOENCACRONICATypeCODCRONICA();


                    idPac.setValue( new BigInteger(
rs.getString("ID_PACIENTE") ) );
                    idCron.setValue( new BigInteger(
rs.getString("ID_CRONICAS") ) );
                    tabCron.setValue( rs.getString("TAB_DIAGNOSTICO")
);
                    codCronica.setValue( rs.getString("COD_CRONICA")
);

                    cronicaList.add(idCron);
                    cronicaList.add(idPac);
                    cronicaList.add(tabCron);
                    cronicaList.add(codCronica);

                }


                 //paList.add(cronicaList);
                pa.getIDENTIFICACAOAndENDERECOAndCONTATO().add(5 ,
cronicaList);


                Marshaller m = jc.createMarshaller();
                StringWriter swriter = new StringWriter();
```

```
                m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
Boolean.TRUE);
                m.marshal(pa,swriter);

                xmlStr = swriter.toString();



            }else
            {
                return "1|Paciente nÃ£o Cadastrado";
            }


        }
        catch(JAXBException ex)
        {
           return "1|JAXBException= " + ex.getMessage();
        }
        catch(Exception ex)
        {
           return "1|Exception= " + ex.getMessage();
        }

        return "0|" + xmlStr;

    }



}
```

Classe: InserirDadosAtendimento.

```
/*
 * InserirDadosAtendimento.java
 *
 * Created on 29 de Outubro de 2005, 14:56
 *
 * To change this template, choose Tools | Options and locate the
template under
 * the Source Creation and Management node. Right-click the template
and choose
 * Open. You can then make changes to the template in the Source
Editor.
 */

package org.anderson.prontuario.atendimento;
import java.io.*;
import javax.xml.transform.stream.*;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Unmarshaller;
import java.util.List;
import java.util.Date;
import java.util.Calendar;
import java.text.SimpleDateFormat;
import java.sql.ResultSet;
import org.anderson.prontuario.banco.Banco;
```

```
/**
 *
 * @author anderson
 */
public class InserirDadosAtendimento {
    private JAXBContext jc;
    private Unmarshaller u;
    private StringBuffer xmlStr;
    private String dados;
    private Banco BD;
    private String SQL="";
    private String resultado="0|";


    /** Creates a new instance of InserirDadosAtendimento */
    public InserirDadosAtendimento(String docXML) {
        try
         {
            jc =
JAXBContext.newInstance("org.anderson.prontuario.atendimento");
            u = jc.createUnmarshaller();
            u.setValidating(true);
            xmlStr = new StringBuffer(docXML);
            BD = new Banco();
         }
         catch(JAXBException je)
         {
                je.printStackTrace();

         }
         catch(Exception ex)
         {
             ex.printStackTrace();
         }

    }


    public String insereDados()
    {   int i;
        ResultSet rs,rsAux=null;
        int idAtendimento;

        String result = BD.criarConexao();
        if( result != "0")
            return "1|Erro ao conectar-se a Base de Dados: ParÃ¢metros
de conexÃ£o Incorretos!";


        try
        {
             ATENDIMENTO atendimento  = (ATENDIMENTO) u.unmarshal(new
StreamSource(new StringReader(xmlStr.toString())));
             List atendimentoList =
atendimento.getCODIGOSAndEXAMEAndEUMA();

             //Buscar ID para PROFISSIONAL
             SQL = "SELECT MAX(ID_ATENDIMENTO) FROM
EVENTO_ATENDIMENTO";

             rs = BD.executarSQL(SQL);
```

```
            if(rs != null)
            {
                rs.next();
                idAtendimento = rs.getInt(1) + 1;
            }
            else
                idAtendimento = 1;

           for( i = 0; i+1 <atendimentoList.size();i+=4 ) {


               ATENDIMENTO.CODIGOS codigos = (ATENDIMENTO.CODIGOS)
atendimentoList.get(i);
               ATENDIMENTO.EXAME exame = (ATENDIMENTO.EXAME)
atendimentoList.get(i+1);
               ATENDIMENTO.EUMA euma = (ATENDIMENTO.EUMA)
atendimentoList.get(i+2);
               ATENDIMENTO.PROCEDIMENTO procedimento =
(ATENDIMENTO.PROCEDIMENTO) atendimentoList.get(i+3);

               //Verifica se codigos informados conferem
               rsAux = null;
               SQL = "SELECT ID_FONTE_PAGADORA FROM FONTE_PAGADORA
WHERE ID_FONTE_PAGADORA = " + codigos.getIDFONTEPAGADORA();
               rsAux = BD.executarSQL(SQL);

               if(rsAux == null)
                   return "1|CÃ³digo de Fonte pagadora invÃ¡lido!";

               rsAux = null;
               SQL = "SELECT ID_PACIENTE FROM IDENTIFICACAO_PACIENTE
WHERE ID_PACIENTE = " + codigos.getIDPACIENTE();
               rsAux = BD.executarSQL(SQL);

               if(rsAux == null)
                   return "1|CÃ³digo do Paciente invÃ¡lido!";

               rsAux = null;
               SQL = "SELECT ID_PRESTADOR FROM PRESTADORA_ASSISTENCIA
WHERE ID_PRESTADOR = " + codigos.getIDPRESTADOR();
               rsAux = BD.executarSQL(SQL);

               if(rsAux == null)
                   return "1|CÃ³digo da Prestadora de Assistencia
invÃ¡lido!";

               //Verificar a existencia de um relacionamento entre
Paceinte e Prestador
               //Se nÃ£o houver insere
               rsAux = null;
               SQL = "SELECT ID_PRESTADOR, ID_PACIENTE  FROM
PACIENTE_PRESTADOR WHERE ID_PRESTADOR = " + codigos.getIDPRESTADOR()
                           +" AND ID_PACIENTE = " +
codigos.getIDPACIENTE();

               rsAux = BD.executarSQL(SQL);

               if(rsAux == null)
               {
```

```
                    SQL = "INSERT INTO PACIENTE_PRESTADOR(
ID_PACIENTE, ID_PRESTADOR ) " +
                           "VALUES( " + codigos.getIDPACIENTE() +
                           "," +  codigos.getIDPRESTADOR()     +
                           ")";

                    if( BD.executarUpdate(SQL) != 1)
                     return "1|Ocorreu um erro na inserÃ§Ã£o do
Relacionamento entre Paciente:" + codigos.getIDPACIENTE()
                            + "  e Prestador: " +
codigos.getIDPRESTADOR();

                }


                SQL = "INSERT INTO EVENTO_ATENDIMENTO( ID_ATENDIMENTO,
ID_FONTE_PAGADORA, ID_PACIENTE, ID_PRESTADOR, TIPO_EVENTO )" +
                        "VALUES(" + idAtendimento +
                        " , " + codigos.getIDFONTEPAGADORA() +
                        " , " + codigos.getIDPACIENTE()  +
                        " , " + codigos.getIDPRESTADOR() +
                        " , " + codigos.getTIPOEVENTO() +
                        ")";


                //Verificar se Profissional Ã© VÃ¡lido

                rsAux = null;
                SQL = "SELECT ID_PROFISSIONAL FROM PROFISSIONAL WHERE
ID_PROFISSIONAL = " + codigos.getIDPROFISSIONAL();
                rsAux = BD.executarSQL(SQL);

                if(rsAux == null)
                   return "1|CÃ³digo do Profissional invÃ¡lido!";


                //Inserir Relacionamento entre Atendimento e
profissional
                SQL = "INSERT INTO ATENDIMENTO_PROFISSIONAL(
ID_PROFISSIONAL, ID_PACIENTE, ID_PRESTADOR, ID_ATENDIMENTO,
ID_FONTE_PAGADORA  ) " +
                           "VALUES( " + codigos.getIDPROFISSIONAL() +
                           "," +  codigos.getIDPACIENTE()           +
                           "," +  codigos.getIDPRESTADOR()          +
                           "," +  idAtendimento                     +
                           "," +  codigos.getIDFONTEPAGADORA()      +
                           ")";


                if( BD.executarUpdate(SQL) != 1)
                     return "1|Ocorreu um erro na inserÃ§Ã£o
relacionamento entre ATENDIMENTO e PRESTADOR: " +
                        codigos.getIDPRESTADOR();


                //TIPOS DE EVENTO
                //1 - EXAME
                //2 - DIAGNOSTICO
                //3 - INTERNACAO
                //4 - OBITO
```

```
                if(codigos.getTIPOEVENTO() == 1)//EXAME
                {
                    int idExame;

                    SQL = "SELECT MAX(ID_EXAME) FROM EXAMES WHERE
ID_PACIENTE = "+ codigos.getIDPACIENTE();
                    if(rs != null)
                    {
                        rs.next();
                        idExame = rs.getInt(1) + 1;
                    }
                    else
                        idExame = 1;

                    String data="2005-10-29";

                    SQL = "INSERT INTO EXAMES( ID_EXAME,
ID_PROFISSIONAL, ID_PACIENTE, ID_PRESTADOR, ID_ATENDIMENTO,
ID_FONTE_PAGADORA, "+
                                              " TAB_CODIFICACAO,
COD_EXAME, RESULTADO, DATAHORA_EXAME )" +
                         "VALUES(" + idExame +
                         " , " + codigos.getIDPROFISSIONAL()    +
                         " , " + codigos.getIDPACIENTE()        +
                         " , " + codigos.getIDPRESTADOR()       +
                         " , " + idAtendimento                  +
                         " , " + codigos.getIDFONTEPAGADORA()   +
                         " ,'" + exame.getTABCODIFICACAO()      +
                         "','" + exame.getCODEXAME()            +
                         "','" + exame.getRESULTADO()           +
                         "','" + data                           +
                         "') ";

                    if( BD.executarUpdate(SQL) != 1)
                        return "1|Ocorreu um erro na inserÃ§Ã£o do
EXAME do Paciente: " +
                           codigos.getIDPACIENTE();
                }


                //Ã‰ UMA----------------------------------------------
----------------------------------------------------------
                if(codigos.getTIPOEVENTO() == 2)//DIAGNOSTICO
                {

                    ATENDIMENTO.EUMA.DIAGNOSTICOType diagnostico =
(ATENDIMENTO.EUMA.DIAGNOSTICOType) euma.getDIAGNOSTICO();


                    SQL = "INSERT INTO DIAGNOSTICO ( ID_ATENDIMENTO,
ID_FONTE_PAGADORA, ID_PACIENTE, ID_PRESTADOR, "+
                                              " TAB_DIAGNOSTICO,
COD_DIAGNOSTICO, COD_SECUNDARIO )" +
                         "VALUES(" + idAtendimento +
                         " , " + codigos.getIDFONTEPAGADORA()       +
                         " , " + codigos.getIDPACIENTE()            +
                         " ,'" + diagnostico.getTABDIAGNOSTICO()    +
                         "','" + diagnostico.getCODDIAGNOSTICO()    +
                         "','" + diagnostico.getCODSECUNDARIO()     +
                         "') ";
```

```
                    if( BD.executarUpdate(SQL) != 1)
                        return "1|Ocorreu um erro na inserÃ§Ã£o do
Diagnostico do Paciente:  " +
                          codigos.getIDPACIENTE();
                }


                if(codigos.getTIPOEVENTO() == 3)//INTERNAÃ‡Ãƒ O
                {

                    ATENDIMENTO.EUMA.INTERNACAOType internacao =
(ATENDIMENTO.EUMA.INTERNACAOType) euma.getINTERNACAO();


                    SQL = "INSERT INTO INTERNACAO( ID_ATENDIMENTO,
ID_FONTE_PAGADORA, ID_PACIENTE, ID_PRESTADOR, "+
                                                  " DATAHORA_ENTRADA,
DATAHORA_SAIDA, TIPO_SAIDA, CONDICAO_ALTA )" +
                           "VALUES(" + idAtendimento +
                           " , " + codigos.getIDFONTEPAGADORA()       +
                           " , " + codigos.getIDPACIENTE()            +
                           " ,'" + internacao.getDATAHORAENTRADA().YEAR
+ "-" + internacao.getDATAHORAENTRADA().MONTH + 1 +"-"+
internacao.getDATAHORAENTRADA().DAY_OF_MONTH  +
                           "','" + internacao.getDATAHORASAIDA().YEAR +
"-" + internacao.getDATAHORASAIDA().MONTH + 1 +"-"+
internacao.getDATAHORASAIDA().DAY_OF_MONTH  +
                           "','" + internacao.getTIPOSAIDA()          +
                           "','" + internacao.getCONDICAOALTA()       +
                           "') ";

                    if( BD.executarUpdate(SQL) != 1)
                        return "1|Ocorreu um erro na inserÃ§Ã£o da
InternaÃ§Ã£o do Paciente:  " +
                          codigos.getIDPACIENTE();

                }


                if(codigos.getTIPOEVENTO() == 4)//OBITO
                {

                    ATENDIMENTO.EUMA.OBITOType obito =
(ATENDIMENTO.EUMA.OBITOType) euma.getOBITO();

                    SQL = "INSERT INTO OBITO( ID_ATENDIMENTO,
ID_FONTE_PAGADORA, ID_PACIENTE, ID_PRESTADOR, "+
                                                  " DATAHORA_OBITO,
TAB_DIAGNOSTICO, COD_DIAGNOSTICO, ENCAMINHADO_NECROPSIA )" +
                           "VALUES(" + idAtendimento +
                           " , " + codigos.getIDFONTEPAGADORA()       +
                           " , " + codigos.getIDPACIENTE()            +
                           " ,'" + obito.getDATAHORAOBITO().YEAR + "-" +
obito.getDATAHORAOBITO().MONTH + 1 +"-"+
obito.getDATAHORAOBITO().DAY_OF_MONTH  +
                           "','" + obito.getTABDIAGNOSTICO()          +
                           "','" + " " +
                           "','" + " " +
                           "') ";

                    if( BD.executarUpdate(SQL) != 1)
```

```
                        return "1|Ocorreu um erro na inserÃ§Ã£o do
Obito do Paciente:  " +
                            codigos.getIDPACIENTE();

                }

               //-------------------------------------------------
-------------------------------------------------
                int idProcedimento;
                SQL = "SELECT MAX(ID_PROCEDIMENTO) FROM PROCEDIMENTO";
                if(rs != null)
                {
                    rs.next();
                    idProcedimento = rs.getInt(1) + 1;
                }
                else
                    idProcedimento = 1;


               //INSERIR PROCEDIMENTO
               //SEMPRE SERÃ  ATRIBUIDO UM PROCEDIMENTO PARA O
ATENDIMENTO
               SQL = "INSERT INTO PROCEDIMENTO( ID_PROCEDIMENTO,
ID_FONTE_PAGADORA, ID_ATENDIMENTO, ID_PROFISSIONAL, ID_PACIENTE,
ID_PRESTADOR, "+
                     " TAB_CODIFICACAO, COD_PROCEDIMENTO,
DATAHORA_PROCEDIMENTO )" +
                     "VALUES(" + idProcedimento                 +
                     " , " + codigos.getIDFONTEPAGADORA()       +
                     " , " + idAtendimento                      +
                     " , " + codigos.getIDPROFISSIONAL()        +
                     " , " + codigos.getIDPACIENTE()            +
                     " , " + codigos.getIDPRESTADOR()           +
                     " ,'" + procedimento.getTABCODIFICACAO()   +
                     "','" + procedimento.getCODPROCEDIMENTO()  +
                     "','" +
procedimento.getDATAHORAPROCEDIMENTO().YEAR + "-" +
procedimento.getDATAHORAPROCEDIMENTO().MONTH + 1 +"-"+
procedimento.getDATAHORAPROCEDIMENTO().DAY_OF_MONTH  +
                     "') ";

               if( BD.executarUpdate(SQL) != 1)
                        return "1|Ocorreu um erro na inserÃ§Ã£o do
Procedimento:  " +
                            codigos.getIDPACIENTE();

               //Retorna cÃ³digo do Profissional indicando sucesso
               resultado+= idAtendimento+"|";
               idAtendimento++;

            }


        }
        catch(JAXBException je)
        {
           return "1|JAXBException je= " + je.getMessage();
        }
        catch(ClassCastException ex)
        {
```

```
           return "1|ClassCastException = " + ex.getClass() +"\n"+
ex.getCause() +"\n"+ ex.getMessage();
        }
        catch(Exception ex)
        {
            return "1|Outras exceÃ§Ãµes= " + ex.getMessage();
        }
        finally{
            BD.desconecta();
        }
        return resultado;
    }
}
```

Classe: BuscarDadosAtendimento.

```
/*
 * BuscarDadosAtendimento.java
 *
 * Created on 29 de Outubro de 2005, 14:57
 *
 * To change this template, choose Tools | Options and locate the
template under
 * the Source Creation and Management node. Right-click the template
and choose
 * Open. You can then make changes to the template in the Source
Editor.
 */

package org.anderson.prontuario.atendimento;
import javax.xml.transform.dom.DOMResult;
import javax.xml.transform.stream.*;
import javax.xml.bind.JAXBContext;
import javax.xml.bind.JAXBException;
import javax.xml.bind.Marshaller;
import java.util.List;
import java.util.Calendar;
import java.math.BigInteger;
import java.sql.ResultSet;
import org.anderson.prontuario.banco.Banco;
import java.io.*;
/**
 *
 * @author anderson
 */
public class BuscarDadosAtendimento {
private Banco BD;
private String SQL;
private ResultSet rs;
private String xmlStr;

    /** Creates a new instance of BuscarDadosAtendimento */
    public BuscarDadosAtendimento() {
        BD = new Banco();
    }

    public String buscarAtendimento(long idAtendimento){
        try
        {
```

```
           JAXBContext jc =
JAXBContext.newInstance("org.anderson.prontuario.atendimento");
           ObjectFactory ofact = new ObjectFactory();
           ATENDIMENTO atendimento = ofact.createATENDIMENTO();

           BD.criarConexao();

           SQL = "SELECT * FROM EVENTO_ATENDIMENTO " +
                    " WHERE ID_ATENDIMENTO = " + idAtendimento;
           rs = BD.executarSQL(SQL);




           if(rs.next())
           {
               //ID_ATENDIMENTO, ID_FONTE_PAGADORA, ID_PACIENTE,
ID_PRESTADOR, TIPO_EVENTO

                ATENDIMENTO.CODIGOS codigos =
ofact.createATENDIMENTOTypeCODIGOS();

                codigos.setIDATENDIMENTO( new
BigInteger(rs.getString("ID_ATENDIMENTO") ));
                codigos.setIDFONTEPAGADORA( new
BigInteger(rs.getString("ID_FONTE_PAGADORA") ));
                codigos.setIDPACIENTE( new
BigInteger(rs.getString("ID_PACIENTE") ));
                codigos.setIDPRESTADOR( new
BigInteger(rs.getString("ID_PRESTADOR") ));
                codigos.setTIPOEVENTO( rs.getInt("TIPO_EVENTO") );



                //TIPOS DE EVENTO
                //1 - EXAME
                //2 - DIAGNOSTICO
                //3 - INTERNACAO
                //4 - OBITO
                //


                ATENDIMENTO.EXAME exame =
ofact.createATENDIMENTOTypeEXAME();

                if (rs.getInt("TIPO_EVENTO") == 1){//EXAME

                    ResultSet rsExame;


                    SQL = "SELECT * FROM EXAMES " +
                        "WHERE ID_ATENDIMENTO = " + idAtendimento;
                    rsExame = BD.executarSQL(SQL);

                    if (rsExame.next()){

                    exame.setIDEXAME(new
BigInteger(rsExame.getString("ID_EXAME")));
                    codigos.setIDPROFISSIONAL(new
BigInteger(rsExame.getString("ID_PROFISSIONAL")));
```

```
                    exame.setIDEXAME(new
BigInteger(rsExame.getString("ID_EXAME")));
                    exame.setTABCODIFICACAO(
rsExame.getString("TAB_CODIFICACAO"));
                    exame.setCODEXAME(rsExame.getString("COD_EXAME"));

exame.setRESULTADO(rsExame.getString("RESULTADO"));
                    exame.setDATAHORAEXAME(Calendar.getInstance());


                    }


                }

                atendimento.getCODIGOSAndEXAMEAndEUMA().add(1, exame);

                ATENDIMENTO.EUMA euma =
ofact.createATENDIMENTOTypeEUMA();



               if (rs.getInt("TIPO_EVENTO") == 2){//DIAGNOSTICO
                  ResultSet rsDiag;



                  SQL = "SELECT * FROM DIAGNOSTICO " +
                    "WHERE ID_ATENDIMENTO = " + idAtendimento;

                  rsDiag = BD.executarSQL(SQL);

                  ATENDIMENTO.EUMA.DIAGNOSTICOType diagnostico =
ofact.createATENDIMENTOTypeEUMATypeDIAGNOSTICOType();

                  if (rsDiag.next()){


diagnostico.setTABDIAGNOSTICO(rsDiag.getString("TAB_DIAGNOSTICO"));

diagnostico.setCODDIAGNOSTICO(rsDiag.getString("COD_DIAGNOSTICO"));

diagnostico.setCODSECUNDARIO(rsDiag.getString("COD_SECUNDARIO"));

                  }

                  euma.setDIAGNOSTICO(diagnostico);

                }


                if (rs.getInt("TIPO_EVENTO") == 3){//INTERNAÃ‡ÃƒO
                  ResultSet rsInt;


                  SQL = "SELECT * FROM INTERNACAO " +
                    "WHERE ID_ATENDIMENTO = " + idAtendimento;

                  rsInt = BD.executarSQL(SQL);
```

```
                    ATENDIMENTO.EUMA.INTERNACAOType internacao =
ofact.createATENDIMENTOTypeEUMATypeINTERNACAOType();

                    if (rsInt.next()){


internacao.setDATAHORAENTRADA(Calendar.getInstance());

internacao.setDATAHORASAIDA(Calendar.getInstance());

internacao.setTIPOSAIDA(rsInt.getString("TIPO_SAIDA"));

internacao.setCONDICAOALTA(rsInt.getString("CONDICAO_ALTA"));

                    }

                    euma.setINTERNACAO(internacao);

                  }


                  if (rs.getInt("TIPO_EVENTO") == 4){//OBITO
                    ResultSet rsObito;


                    SQL = "SELECT * FROM OBITO " +
                      "WHERE ID_ATENDIMENTO = " + idAtendimento;

                    rsObito = BD.executarSQL(SQL);

                    ATENDIMENTO.EUMA.OBITOType obito =
ofact.createATENDIMENTOTypeEUMATypeOBITOType();

                    if (rsObito.next()){

                       obito.setDATAHORAOBITO(Calendar.getInstance());

obito.setTABDIAGNOSTICO(rsObito.getString("TAB_DIAGNOSTICO"));

                    }

                    euma.setOBITO(obito);

                  }

                  atendimento.getCODIGOSAndEXAMEAndEUMA().add(2, euma);



                   // Procedimento

                  ATENDIMENTO.PROCEDIMENTO procedimento =
ofact.createATENDIMENTOTypePROCEDIMENTO();

                  ResultSet rsProc;


                  SQL = "SELECT * FROM EXAMES " +
                          "WHERE ID_ATENDIMENTO = " + idAtendimento;

                  rsProc = BD.executarSQL(SQL);
```

```
                if (rsProc.next()){

                    procedimento.setIDPROCEDIMENTO(new
BigInteger(rsProc.getString("ID_PROCEDIMENTO")) );

procedimento.setCODPROCEDIMENTO(rsProc.getString("COD_PROCEDIMENTO")
);

procedimento.setTABCODIFICACAO(rsProc.getString("TAB_CODIFICACAO") );

procedimento.setDATAHORAPROCEDIMENTO(Calendar.getInstance());

                }

                atendimento.getCODIGOSAndEXAMEAndEUMA().add(3,
procedimento);

                Marshaller m = jc.createMarshaller();
                StringWriter swriter = new StringWriter();

                m.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
Boolean.TRUE);
                m.marshal(atendimento,swriter);

                xmlStr = swriter.toString();


           }else
               return "1|Evento ou Atendimento nÃ£o encontrado";


        }
        catch(JAXBException ex)
        {
           return "1|JAXBException= " + ex.getMessage();
        }
        catch(Exception ex)
        {
           return "1|Exception= " + ex.getMessage();
        }

        return "0|" + xmlStr;

    }
}
```

Classe de Acesso a Base da Dados:

Classe: Banco.

```
/*
 * Banco.java
 *
 * Created on 10 de Setembro de 2005, 14:29
 *
 * To change this template, choose Tools | Options and locate the
template under
```

```
 * the Source Creation and Management node. Right-click the template
and choose
 * Open. You can then make changes to the template in the Source
Editor.
 */

package org.anderson.prontuario.banco;
import java.sql.*;
import com.mysql.jdbc.Driver;

/**
 *
 * @author anderson
 */
public class Banco {
    private Connection connection = null;
    private Statement statement = null;
    private ResultSet rs;
    private int numElementos = 0;

    private String drivename;//com.mysql.jdbc.Driver
    private String username; //root
    private String password;
    private String databaseurl;
//jdbc:mysql://localhost:3306/PRONTUARIO


    /** Creates a new instance of Banco */
    public Banco() {
        this.drivename = "com.mysql.jdbc.Driver";
        this.username = "root";
        this.password = "";
        this.databaseurl = "jdbc:mysql://localhost:3306/PRONTUARIO";
    }

    public Banco(String drivename, String username, String password,
String databaseurl){
        this.drivename = drivename;
        this.username = username;
        this.password = password;
        this.databaseurl = databaseurl;

    }


    public String criarConexao(){
        try{
            Class.forName("com.mysql.jdbc.Driver").newInstance();
            connection =
DriverManager.getConnection(databaseurl,username,password);
            statement = connection.createStatement();
            return "0";

        }
        catch(ClassNotFoundException exception)
        {
           return "ClassNotFoundException : " +
exception.getMessage();
        }
        catch(SQLException exception)
        {
```

```
           return "SQLException : " +  exception.getMessage();
        }
        catch(InstantiationException exception)
        {
           return "InstantiationException : " +
exception.getMessage();
        }
        catch(Exception exception)
        {
            return "Exception: " +  exception.getMessage();
        }



    }



  public int executarUpdate(String sql) {
       int i=3;
        try
        {
             i = statement.executeUpdate(sql);
             return  i;
             // 1 -  Sucesso - E retorna numero de linha afetadas
             // 2 -  Sucesso - porem nenhuma linha afetada
             // 3 -  Falha de ExecuÃ§Ã£o
        }
        catch(SQLException exception)
        {
           String msgErro = exception.getMessage();
           return i;
        }

    }


   public ResultSet executarSQL(String sql){
       try
       {
           rs = statement.executeQuery(sql);
           return rs;
       }
       catch(SQLException exception)
       {   rs = null;
           return rs;
       }

   }



    public boolean desconecta(){
        try{
            statement.close();
            connection.close();
            return true;
        }
        catch(SQLException ex){
            ex.printStackTrace();
            return false;
```

```
        }

    }

}
```

Classes Geradas pelo Compilador de Ligação da Arquitetura JAXB.

Observação: Será mostrada apenas as classes geradas através do documento PrestadoraAssistencia.xsd.

Classe: Interface PRESTADOR.

```
//
// This file was generated by the JavaTM Architecture for XML
Binding(JAXB) Reference Implementation, v1.0.4-b18-fcs
// See <a
href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jaxb</a>
// Any modifications to this file will be lost upon recompilation of
the source schema.
// Generated on: 2005.09.09 at 10:21:14 BRT
//


package org.anderson.prontuario.prestadora;


/**
 * Comment describing your root element
 * Java content class for PRESTADOR element declaration.
 * <p>The following schema fragment specifies the expected content
contained within this java content object. (defined at
file:/home/anderson/XML_PACIENTE/Prestadora_Assistencia.xsd line 4)
 * <p>
 * <pre>
 * &lt;element name="PRESTADOR">
 *   &lt;complexType>
 *     &lt;complexContent>
 *       &lt;restriction
base="{http://www.w3.org/2001/XMLSchema}anyType">
 *         &lt;sequence maxOccurs="unbounded">
 *           &lt;element name="ID_PRESTADOR"
type="{http://www.w3.org/2001/XMLSchema}unsignedLong" minOccurs="0"/>
 *           &lt;element name="NOME_INSTITUICAO">
 *             &lt;restriction
base="{http://www.w3.org/2001/XMLSchema}string">
 *               &lt;minLength value="1"/>
 *               &lt;maxLength value="50"/>
 *             &lt;/restriction>
 *           &lt;/element>
 *           &lt;element name="CNPJ_INSTITUICAO">
 *             &lt;restriction
base="{http://www.w3.org/2001/XMLSchema}string">
 *               &lt;minLength value="1"/>
 *               &lt;maxLength value="18"/>
 *             &lt;/restriction>
 *           &lt;/element>
 *         &lt;/sequence>
```

```
 *       &lt;/restriction>
 *     &lt;/complexContent>
 *   &lt;/complexType>
 * &lt;/element>
 * </pre>
 *
 */
public interface PRESTADOR
    extends javax.xml.bind.Element,
org.anderson.prontuario.prestadora.PRESTADORType
{


}
```

Classe: Interface PRESTADORType.

```
//
// This file was generated by the JavaTM Architecture for XML
Binding(JAXB) Reference Implementation, v1.0.4-b18-fcs
// See <a
href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jaxb</a>
// Any modifications to this file will be lost upon recompilation of
the source schema.
// Generated on: 2005.09.09 at 10:21:14 BRT
//


package org.anderson.prontuario.prestadora;


/**
 * Java content class for anonymous complex type.
 * <p>The following schema fragment specifies the expected content
contained within this java content object. (defined at
file:/home/anderson/XML_PACIENTE/Prestadora_Assistencia.xsd line 8)
 * <p>
 * <pre>
 * &lt;complexType>
 *   &lt;complexContent>
 *     &lt;restriction
base="{http://www.w3.org/2001/XMLSchema}anyType">
 *       &lt;sequence maxOccurs="unbounded">
 *         &lt;element name="ID_PRESTADOR"
type="{http://www.w3.org/2001/XMLSchema}unsignedLong" minOccurs="0"/>
 *         &lt;element name="NOME_INSTITUICAO">
 *           &lt;restriction
base="{http://www.w3.org/2001/XMLSchema}string">
 *             &lt;minLength value="1"/>
 *             &lt;maxLength value="50"/>
 *           &lt;/restriction>
 *         &lt;/element>
 *         &lt;element name="CNPJ_INSTITUICAO">
 *           &lt;restriction
base="{http://www.w3.org/2001/XMLSchema}string">
 *             &lt;minLength value="1"/>
 *             &lt;maxLength value="18"/>
 *           &lt;/restriction>
 *         &lt;/element>
 *       &lt;/sequence>
```

```
 *     &lt;/restriction>
 *   &lt;/complexContent>
 * &lt;/complexType>
 * </pre>
 *
 */
public interface PRESTADORType {


    /**
     * Gets the value of the
IDPRESTADORAndNOMEINSTITUICAOAndCNPJINSTITUICAO property.
     *
     * <p>
     * This accessor method returns a reference to the live list,
     * not a snapshot. Therefore any modification you make to the
     * returned list will be present inside the JAXB object.
     * This is why there is not a <CODE>set</CODE> method for the
IDPRESTADORAndNOMEINSTITUICAOAndCNPJINSTITUICAO property.
     *
     * <p>
     * For example, to add a new item, do as follows:
     * <pre>
     *
getIDPRESTADORAndNOMEINSTITUICAOAndCNPJINSTITUICAO().add(newItem);
     * </pre>
     *
     *
     * <p>
     * Objects of the following type(s) are allowed in the list
     * {@link
org.anderson.prontuario.prestadora.PRESTADORType.IDPRESTADOR}
     * {@link
org.anderson.prontuario.prestadora.PRESTADORType.NOMEINSTITUICAO}
     * {@link
org.anderson.prontuario.prestadora.PRESTADORType.CNPJINSTITUICAO}
     *
     */
    java.util.List
getIDPRESTADORAndNOMEINSTITUICAOAndCNPJINSTITUICAO();


    /**
     * Java content class for CNPJ_INSTITUICAO element declaration.
     * <p>The following schema fragment specifies the expected content
contained within this java content object. (defined at
file:/home/anderson/XML_PACIENTE/Prestadora_Assistencia.xsd line 19)
     * <p>
     * <pre>
     * &lt;element name="CNPJ_INSTITUICAO">
     *   &lt;restriction
base="{http://www.w3.org/2001/XMLSchema}string">
     *     &lt;minLength value="1"/>
     *     &lt;maxLength value="18"/>
     *   &lt;/restriction>
     * &lt;/element>
     * </pre>
     *
     */
    public interface CNPJINSTITUICAO
        extends javax.xml.bind.Element
```

```
    {


        /**
         * Gets the value of the value property.
         *
         * @return
         *     possible object is
         *     {@link java.lang.String}
         */
        java.lang.String getValue();

        /**
         * Sets the value of the value property.
         *
         * @param value
         *     allowed object is
         *     {@link java.lang.String}
         */
        void setValue(java.lang.String value);

    }


    /**
     * Java content class for ID_PRESTADOR element declaration.
     * <p>The following schema fragment specifies the expected content
contained within this java content object. (defined at
file:/home/anderson/XML_PACIENTE/Prestadora_Assistencia.xsd line 10)
     * <p>
     * <pre>
     * &lt;element name="ID_PRESTADOR"
type="{http://www.w3.org/2001/XMLSchema}unsignedLong"/>
     * </pre>
     *
     */
    public interface IDPRESTADOR
        extends javax.xml.bind.Element
    {


        /**
         * Gets the value of the value property.
         *
         * @return
         *     possible object is
         *     {@link java.math.BigInteger}
         */
        java.math.BigInteger getValue();

        /**
         * Sets the value of the value property.
         *
         * @param value
         *     allowed object is
         *     {@link java.math.BigInteger}
         */
        void setValue(java.math.BigInteger value);

    }
```

```
    /**
     * Java content class for NOME_INSTITUICAO element declaration.
     * <p>The following schema fragment specifies the expected content
contained within this java content object. (defined at
file:/home/anderson/XML_PACIENTE/Prestadora_Assistencia.xsd line 11)
     * <p>
     * <pre>
     * &lt;element name="NOME_INSTITUICAO">
     *   &lt;restriction
base="{http://www.w3.org/2001/XMLSchema}string">
     *     &lt;minLength value="1"/>
     *     &lt;maxLength value="50"/>
     *   &lt;/restriction>
     * &lt;/element>
     * </pre>
     *
     */
    public interface NOMEINSTITUICAO
        extends javax.xml.bind.Element
    {


        /**
         * Gets the value of the value property.
         *
         * @return
         *     possible object is
         *     {@link java.lang.String}
         */
        java.lang.String getValue();

        /**
         * Sets the value of the value property.
         *
         * @param value
         *     allowed object is
         *     {@link java.lang.String}
         */
        void setValue(java.lang.String value);

    }

}
```

Classe: ObjectFactory.

```
//
// This file was generated by the JavaTM Architecture for XML
Binding(JAXB) Reference Implementation, v1.0.4-b18-fcs
// See <a
href="http://java.sun.com/xml/jaxb">http://java.sun.com/xml/jaxb</a>
// Any modifications to this file will be lost upon recompilation of
the source schema.
// Generated on: 2005.09.09 at 10:21:14 BRT
//


package org.anderson.prontuario.prestadora;
```

```
/**
 * This object contains factory methods for each
 * Java content interface and Java element interface
 * generated in the org.anderson.prontuario.prestadora package.
 * <p>An ObjectFactory allows you to programatically
 * construct new instances of the Java representation
 * for XML content. The Java representation of XML
 * content can consist of schema derived interfaces
 * and classes representing the binding of schema
 * type definitions, element declarations and model
 * groups.  Factory methods for each of these are
 * provided in this class.
 *
 */
public class ObjectFactory
    extends
org.anderson.prontuario.prestadora.impl.runtime.DefaultJAXBContextImpl
{

    private static java.util.HashMap defaultImplementations = new
java.util.HashMap(16, 0.75F);
    private static java.util.HashMap rootTagMap = new
java.util.HashMap();
    public final static
org.anderson.prontuario.prestadora.impl.runtime.GrammarInfo
grammarInfo = new
org.anderson.prontuario.prestadora.impl.runtime.GrammarInfoImpl(rootTa
gMap, defaultImplementations,
(org.anderson.prontuario.prestadora.ObjectFactory.class));
    public final static java.lang.Class version =
(org.anderson.prontuario.prestadora.impl.JAXBVersion.class);

    static {

defaultImplementations.put((org.anderson.prontuario.prestadora.PRESTAD
ORType.IDPRESTADOR.class),
"org.anderson.prontuario.prestadora.impl.PRESTADORTypeImpl.IDPRESTADOR
Impl");

defaultImplementations.put((org.anderson.prontuario.prestadora.PRESTAD
ORType.class),
"org.anderson.prontuario.prestadora.impl.PRESTADORTypeImpl");

defaultImplementations.put((org.anderson.prontuario.prestadora.PRESTAD
OR.class), "org.anderson.prontuario.prestadora.impl.PRESTADORImpl");

defaultImplementations.put((org.anderson.prontuario.prestadora.PRESTAD
ORType.CNPJINSTITUICAO.class),
"org.anderson.prontuario.prestadora.impl.PRESTADORTypeImpl.CNPJINSTITU
ICAOImpl");

defaultImplementations.put((org.anderson.prontuario.prestadora.PRESTAD
ORType.NOMEINSTITUICAO.class),
"org.anderson.prontuario.prestadora.impl.PRESTADORTypeImpl.NOMEINSTITU
ICAOImpl");
        rootTagMap.put(new
javax.xml.namespace.QName("http://localhost:8080/ProntuarioService/Sch
ema", "PRESTADOR"),
(org.anderson.prontuario.prestadora.PRESTADOR.class));
```

```
    }

    /**
     * Create a new ObjectFactory that can be used to create new
instances of schema derived classes for package:
org.anderson.prontuario.prestadora
     *
     */
    public ObjectFactory() {
        super(grammarInfo);
    }

    /**
     * Create an instance of the specified Java content interface.
     *
     * @param javaContentInterface
     *     the Class object of the javacontent interface to
instantiate
     * @return
     *     a new instance
     * @throws JAXBException
     *     if an error occurs
     */
    public java.lang.Object newInstance(java.lang.Class
javaContentInterface)
        throws javax.xml.bind.JAXBException
    {
        return super.newInstance(javaContentInterface);
    }

    /**
     * Get the specified property. This method can only be
     * used to get provider specific properties.
     * Attempting to get an undefined property will result
     * in a PropertyException being thrown.
     *
     * @param name
     *     the name of the property to retrieve
     * @return
     *     the value of the requested property
     * @throws PropertyException
     *     when there is an error retrieving the given property or
value
     */
    public java.lang.Object getProperty(java.lang.String name)
        throws javax.xml.bind.PropertyException
    {
        return super.getProperty(name);
    }

    /**
     * Set the specified property. This method can only be
     * used to set provider specific properties.
     * Attempting to set an undefined property will result
     * in a PropertyException being thrown.
     *
     * @param value
     *     the value of the property to be set
     * @param name
     *     the name of the property to retrieve
     * @throws PropertyException
```

```
     *     when there is an error processing the given property or
value
     */
    public void setProperty(java.lang.String name, java.lang.Object
value)
        throws javax.xml.bind.PropertyException
    {
        super.setProperty(name, value);
    }

    /**
     * Create an instance of PRESTADORTypeIDPRESTADOR
     *
     * @throws JAXBException
     *     if an error occurs
     */
    public
org.anderson.prontuario.prestadora.PRESTADORType.IDPRESTADOR
createPRESTADORTypeIDPRESTADOR()
        throws javax.xml.bind.JAXBException
    {
        return new
org.anderson.prontuario.prestadora.impl.PRESTADORTypeImpl.IDPRESTADORI
mpl();
    }

    /**
     * Create an instance of PRESTADORTypeIDPRESTADOR
     *
     * @throws JAXBException
     *     if an error occurs
     */
    public
org.anderson.prontuario.prestadora.PRESTADORType.IDPRESTADOR
createPRESTADORTypeIDPRESTADOR(java.math.BigInteger value)
        throws javax.xml.bind.JAXBException
    {
        return new
org.anderson.prontuario.prestadora.impl.PRESTADORTypeImpl.IDPRESTADORI
mpl(value);
    }

    /**
     * Create an instance of PRESTADORType
     *
     * @throws JAXBException
     *     if an error occurs
     */
    public org.anderson.prontuario.prestadora.PRESTADORType
createPRESTADORType()
        throws javax.xml.bind.JAXBException
    {
        return new
org.anderson.prontuario.prestadora.impl.PRESTADORTypeImpl();
    }

    /**
     * Create an instance of PRESTADOR
     *
     * @throws JAXBException
     *     if an error occurs
```

```
     */
    public org.anderson.prontuario.prestadora.PRESTADOR
createPRESTADOR()
        throws javax.xml.bind.JAXBException
    {
        return new
org.anderson.prontuario.prestadora.impl.PRESTADORImpl();
    }

    /**
     * Create an instance of PRESTADORTypeCNPJINSTITUICAO
     *
     * @throws JAXBException
     *     if an error occurs
     */
    public
org.anderson.prontuario.prestadora.PRESTADORType.CNPJINSTITUICAO
createPRESTADORTypeCNPJINSTITUICAO()
        throws javax.xml.bind.JAXBException
    {
        return new
org.anderson.prontuario.prestadora.impl.PRESTADORTypeImpl.CNPJINSTITUI
CAOImpl();
    }

    /**
     * Create an instance of PRESTADORTypeCNPJINSTITUICAO
     *
     * @throws JAXBException
     *     if an error occurs
     */
    public
org.anderson.prontuario.prestadora.PRESTADORType.CNPJINSTITUICAO
createPRESTADORTypeCNPJINSTITUICAO(java.lang.String value)
        throws javax.xml.bind.JAXBException
    {
        return new
org.anderson.prontuario.prestadora.impl.PRESTADORTypeImpl.CNPJINSTITUI
CAOImpl(value);
    }

    /**
     * Create an instance of PRESTADORTypeNOMEINSTITUICAO
     *
     * @throws JAXBException
     *     if an error occurs
     */
    public
org.anderson.prontuario.prestadora.PRESTADORType.NOMEINSTITUICAO
createPRESTADORTypeNOMEINSTITUICAO()
        throws javax.xml.bind.JAXBException
    {
        return new
org.anderson.prontuario.prestadora.impl.PRESTADORTypeImpl.NOMEINSTITUI
CAOImpl();
    }

    /**
     * Create an instance of PRESTADORTypeNOMEINSTITUICAO
     *
     * @throws JAXBException
```

```
     *     if an error occurs
     */
    public
org.anderson.prontuario.prestadora.PRESTADORType.NOMEINSTITUICAO
createPRESTADORTypeNOMEINSTITUICAO(java.lang.String value)
        throws javax.xml.bind.JAXBException
    {
        return new
org.anderson.prontuario.prestadora.impl.PRESTADORTypeImpl.NOMEINSTITUI
CAOImpl(value);
    }

}
```