UNIVERSIDADE REGIONAL DE BLUMENAU
CENTRO DE CIÊNCIAS EXATAS E NATURAIS
CURSO DE CIÊNCIAS DA COMPUTAÇÃO
(Bacharelado)

SISTEMA PARA CONSULTAS E ALOCAÇÃO DE RECURSOS UTILIZANDO *WEB SERVICES.*

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO À UNIVERSIDADE REGIONAL DE BLUMENAU PARA A OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA COM NOME EQUIVALENTE NO CURSO DE CIÊNCIAS DA COMPUTAÇÃO — BACHARELADO

ANDERSON ROBERTO GERMANO

BLUMENAU, JUNHO/2003

2003/1-05

SISTEMA PARA CONSULTAS E ALOCAÇÃO DE RECURSOS UTILIZANDO *WEB SERVICES*

ANDERSON ROBERTO GERMANO

ESTE TRABALHO DE CONCLUSÃO DE CURSO, FOI JULGADO ADEQUADO PARA OBTENÇÃO DOS CRÉDITOS NA DISCIPLINA DE TRABALHO DE CONCLUSÃO DE CURSO OBRIGATÓRIA PARA OBTENÇÃO DO TÍTULO DE:

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

Prof. Maurício Capobianco Lopes — Orientador na FURB

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

Prof. Maurício Capobianco Lopes

Prof. Wilson Pedro Carli

Prof. Ricardo Alencar Azambuja

RESUMO

Este trabalho apresenta o desenvolvimento de uma aplicação utilizando uma das novas tecnologias referentes à transferência de informações via Web, denominada *Web Services*. Utilizando esta tecnologia foi desenvolvida uma aplicação que permite que fornecedores do setor de turismo possam disponibilizar seus recursos na *Web* para que agentes de viagens pesquisem, entre seus fornecedores, os recursos que melhor atenderem à necessidade de seus clientes, podendo requisitar a reserva do mesmo.

ABSTRACT

This work presents the development of an application using one of the new technologies about the information transference via web, denominated *Web Services*. Using this technology were developed an application that's enable suppliers of tourism sector be able do share their resources in *Web* for that travel agents research, between their suppliers, the resources that best fit to the needs of their clients, been enabled to require the reserve of the same.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE QUADROS

1 INTRODUÇÃO

A rapidez e exatidão vivenciada em tempos modernos são cada vez mais pré-requisitos essenciais para o bom andamento de um negócio. Para que estes pré-requisitos sejam atendidos é necessário que o cliente e o fornecedor estejam interagindo constantemente, e o tempo de resposta destas interações seja o mais curto possível.

Um exemplo disso seria um turista chegando em uma agência de viagem precisando reservar uma acomodação. Para que o agente de viagem pudesse oferecer-lhe todas as opções de apartamentos e hotéis disponíveis, o agente teria que contactar com cada um de seus fornecedores e pedir que lhe fosse enviado uma lista com todos os serviços disponíveis destes fornecedores. Isto com certeza levaria um tempo considerável até que todos os seus fornecedores respondessem, ou o cliente teria que fazer a reserva sem saber de todas as opções realmente disponíveis.

Em algumas páginas na internet que efetuam reservas "*on-line*", pode-se constatar que em sua maioria a reserva funciona da seguinte maneira: o cliente indica em que localidade deseja encontrar o cômodo e uma lista é apresentada com os hotéis que se enquadram neste pedido, sem ter sido verificado anteriormente se estes hotéis realmente possuem estes cômodos vagos. Este método exige que as reservas sejam feitas com um determinado número de dias de antecedência, para que seja possível a verificação da disponibilidade deste recurso diretamente com o fornecedor escolhido, e após esta verificação ter sido feita, é informado ao cliente se sua reserva foi confirmada ou não.

Assim, propõe-se criar uma "*ligação*" entre o agente de viagens e o fornecedor evitando que ao fazer a pesquisa o agente de viagens receba dados de recursos que não estão disponíveis e também permitindo a criação de maiores variedades de filtros nestas pesquisas. Esses filtros permitirão que o agente de viagens escolha por exemplo, o tipo de cômodo, andar, número do quarto, ou seja, todos os campos que o fornecedor disponibilizar na pesquisa.

Para que esta ligação entre o agente de viagens e o fornecedor seja possível é necessário que ambos estejam atualizando-se constantemente ou que a cada requisição de um agente de viagens, o fornecedor possa responder informando suas disponibilidades.

Assim, somente devem trafegar estas informações para o agente de viagens quando este às requisitar.

A proposta deste trabalho é a construção de um protótipo que possibilite que a informação trafegue, somente quando for requisitada. Para a realização do protótipo será utilizada uma das tecnologias mais recentes no mercado referente a transferências de informações na Web, denominada *Web Services.*

Web Services é uma aplicação publicada, localizada e chamada através da internet. Sua função é de encapsular e contratar funções e objetos remotos oferecidos via um protocolo padrão e conhecido (PEREIRA, 2002).

Um *Web Services* é uma interface acessível pela rede para funcionalidades da aplicação, construída usando padrão das tecnologias da internet (SNELL, 2001).

Os *Web Services* tem a capacidade de combinar, compartilhar, trocar ou se conectar a serviços separados de vários fabricantes e desenvolvedores para formar serviços totalmente novos ou personalizar aplicativos criados dinamicamente para que eles atendam às necessidades do cliente (WATSON, 2002).

Utilizando esta tecnologia, quando um turista for fazer uma reserva de apartamento, por exemplo, o agente de viagem poderá mandar uma mensagem para todos os seus fornecedores, sendo que eles responderão após consultarem as bases de dados, trazendo uma lista de todas as opções que se enquadrarem no pedido.

Para isso, propõe-se criar uma aplicação em que os agentes de viagens possam consultar os serviços disponíveis de seus fornecedores, sendo também necessário disponibilizar um aplicativo no fornecedor para manter atualizadas as informações a serem liberadas para pesquisa. Para o fornecedor que já possua um sistema informatizado será disponibilizado um aplicativo que se adapte à sua base, possibilitando a parametrização dos dados a serem disponibilizados pelos *Web Services.* Essa parametrização será feita informando para o sistema quais os campos que poderão ser liberados para as consultas.

1.1 OBJETIVOS DO TRABALHO

O objetivo do trabalho é criar uma aplicação no ramo de turismo que use a tecnologia *Web Services.*

Os objetivos específicos são:

a) permitir a parametrização pelos fornecedores dos dados a serem disponibilizados para a pesquisa dos agentes de viagens;
b) permitir que os agentes de viagens filtrem as informações extraídas dos fornecedores;
c) possibilitar ao agente de viagens fazer reserva de um recurso do fornecedor, através do envio de um pedido. Posteriormente, uma resposta do fornecedor deverá ser enviada confirmando ou não a alocação do pedido;
d) testar a tecnologia *Web Services* no ambiente Delphi.

1.2 ESTRUTURA DO TRABALHO

O trabalho está organizado da seguinte forma:

No primeiro capítulo é feita a introdução dos aspectos a serem abordados nesta monografia.

No segundo capítulo é abordada a tecnologia *Web Services*, a principal tecnologia que será utilizada no desenvolvimento do protótipo.

No terceiro capítulo são descritos todos os passos utilizados para o desenvolvimento do protótipo.

No quarto capítulo, são apresentadas as conclusões e sugestões.

2 WEB SERVICES

Web Services são aplicações modularizadas que podem ser descritas, publicadas e invocadas sobre uma rede, geralmente a WEB. Ou seja, é uma interface que descreve uma coleção de operações que são acessíveis pela rede através de mensagens em formato *eXtensible Markup Language* (XML), padronizadas. Os *Web Services* permitem uma integração de serviços de maneira rápida e eficiente (KREGER, 2001)

Web Services apresentam uma estrutura arquitetural que permite a comunicação entre aplicações. O uso de tecnologias baseadas em XML para a construção de *Web Services*, permite a utilização de serviços sem que haja necessidade de saber qual a plataforma ou linguagem de programação foi utilizada para sua construção. Um *Web Service* pode ser invocado remotamente ou ser utilizado para compor um novo *Web Service*. Algumas tecnologias empregadas para construí-los, são: *Web Services Description Language* (WSDL), *Simple Object Access Protocol* (SOAP) e *Universal Description, Discovery e Integration* (UDDI). Estas tecnologias são baseadas no *eXtensible Markup Language* (XML).

Os servidores podem descrever seus próprios serviços através da WSDL. Dessa forma, facilmente é possível obter informações sobre os *Web Services* que usarão, tais como: estrutura, métodos e parâmetros exigidos. Isso se torna essencialmente útil quando se está codificando *Web Services* que serão usados por terceiros ou implementando aplicações clientes que usam serviços de outras empresas.

2.1 CONCEITOS

Um *Web Service* é um componente de software independente de implementação e plataforma. Pode ser descrito utilizando-se uma linguagem de descrição de serviços, publicado em um registro e descoberto através de um mecanismo padrão. Por conseguinte, também ser invocado a partir de uma *Application Program Interface* (API) através da rede e sendo composto juntamente com outros serviços.

Os *Web Services* combinam os melhores aspectos do desenvolvimento baseado em componentes na WEB. Assim como componentes de *software*, os *Web Services* representam uma funcionalidade do tipo *caixa preta* que pode ser reutilizada sem a preocupação com a linguagem de programação utilizada (GRAHAM, 2002).

O termo *Web Services* descreve funcionalidades especificas do "negócios" exposto usualmente através de conexões de internet com o propósito de fornecer um caminho para a utilização deste serviço.

A exposição dos serviços envolve:

a) identificar ou definir funções de valor no negócio ou processos;
b) definir uma interface baseada no serviço para os processos;
c) descrever estas interfaces em um formato baseado na WEB.

Para tornar um serviço disponível sobre a WEB normalmente é necessário:

a) publicar a interface do serviço para que este possa ser encontrado e utilizado;
b) aceitar requisições e enviar respostas usando protocolo padrão de mensagens XML;
c) fazer uma ligação entre requisições externas e implementações das funções dos negócios.

Utilizando um documento criado na forma de uma mensagem XML, um programa envia uma requisição para um *Web Service* através da rede e opcionalmente recebe uma resposta, também na forma de um documento XML. O uso do XML torna-se importante por disponibilizar um mecanismo extensível para descrever as mensagens e os seus conteúdos. Assim sendo, padrões de serviços WEB definem o formato da mensagem e mecanismos para publicar e descobrir interfaces *Web Services*.

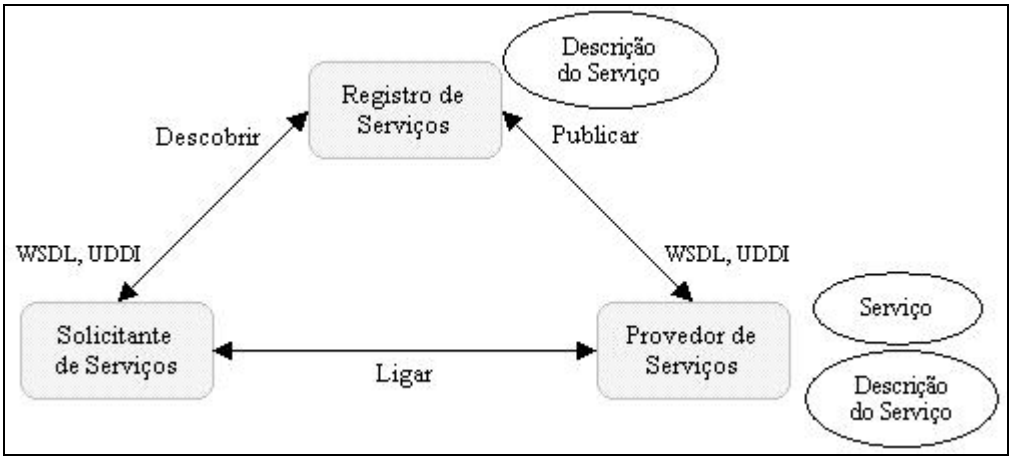
Web Services não são acessados via protocolos específicos de modelos de objetos como, por exemplo, *Distributed Component Object Model* (DCOM), *Remote Method Invocation* (RMI) ou *Internet Inter-ORB Protocol* (IIOP), mas são acessados por protocolos de transporte como *Hiper Text Protocol* (HTTP), FTP, etc.

2.2 ARQUITETURA DE *WEB SERVICES*

Em uma arquitetura *Web Service* a descrição de um serviço cobre todos os detalhes necessários para que haja a interação com um serviço, incluindo o formato de mensagens, os protocolos de transportes e a localização.

A interface esconde os detalhes de implementação do serviço, permitindo que o mesmo possa ser utilizado independentemente da plataforma de *hardware* e *software* e da linguagem de programação utilizadas para o desenvolvimento. Isto permite que as aplicações baseadas em *Web Services* sejam orientadas a componentes e, desta forma, reutilizadas.

Figura 1: *Web Services papéis, operações e ofícios.*



Fonte: KREGER, 2001.

A arquitetura dos *Web Services* apresentada na Figura 1, está baseada nas interações de três funções:

a) **provedor de serviços:** da perspectiva do negócio este é o dono do serviço. Da perspectiva arquitetural esta é a plataforma acessada na solicitação do serviço. É a entidade que cria o *Web Service* sendo responsável por fazer sua descrição em algum formato padrão e publicar os detalhes em um registro central;
b) **solicitante de serviços:** é uma aplicação que invoca ou inicializa uma interação com o serviço. Pode ser um *browser* ou um programa sem interface com o

usuário como por exemplo outro *Web Service*. Através da descrição de um serviço é possível encontrar e invocar os *Web Services*;

c) **registro de serviços:** é o local onde os provedores de serviços publicam suas descrições dos serviços. Os solicitantes de serviços procuram-os e obtêm as informações de ligação da descrição do serviço, durante a fase de desenvolvimento (ligação estática) ou em tempo de execução (ligação dinâmica).

Na arquitetura dos *Web Services* existe também a **Descrição do Serviço** que contém os detalhes da interface e de implementação do serviço, incluindo a estrutura de dados, operações e informações de ligação na rede. Também contém dados para facilitar a sua localização pelo **Solicitante de Serviço**. O **Serviço** é o *software* disponibilizado na rede pelo **Provedor de Serviço.**

Além disso, há algumas operações nesta arquitetura que são:

a) **ligar:** quando um serviço necessita ser utilizado, a operação Ligar invoca e inicializa a interação com o mesmo em tempo de execução, usando as informações das ligações da descrição do serviço para localizar e contatá-lo;
b) **publicar:** para que um serviço possa ser acessado, este deve ser publicado no **Registro de Serviço.** O **Provedor de Serviço** contata o **Registro de Serviço** para publicar um serviço;
c) **descobrir:** um **Solicitante de Serviços** encontra uma descrição do serviço ou consulta o **Registro de Serviço** para o tipo do serviço requerido. Um **Solicitante de Serviços** pode encontrar um descrição da interface do serviço para o desenvolvimento de *software* na fase de *design* ou em tempo de execução. Desta forma, são encontradas as informações sobre ligações e localizações necessárias para invocar um serviço.

2.3 CAMADAS CONCEITUAIS DOS *WEB SERVICES*

A Figura 2 apresenta as camadas conceituais dos *Web Services*, descritas a seguir (KREGER, 2001), (HENDRICKS, 2002):

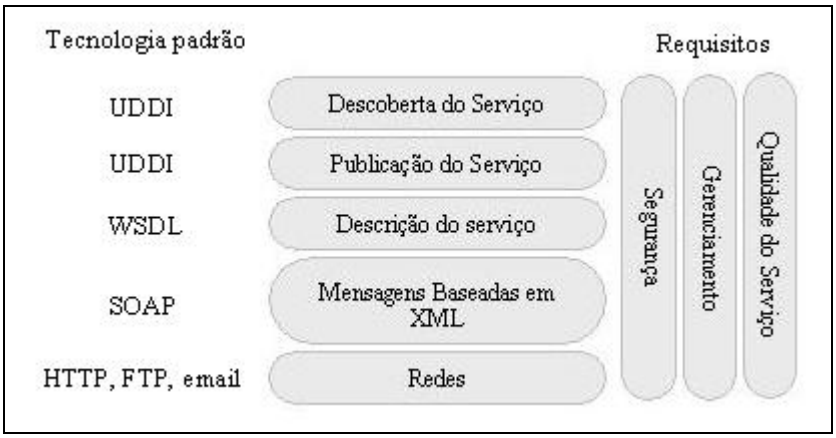
a) **camada de rede:** é a camada base que representa protocolos como: HTTP, FTP, SMTP, POP, etc. Esta camada é utilizada de acordo com os requisitos da aplicação: segurança, disponibilidade, performance e confiabilidade;

b) **mensagens baseada em XML:** esta camada usa como padrão de tecnologia o protocolo SOAP, para troca de informações em um ambiente descentralizado e distribuído;

c) **descrição do Serviço**: a descrição do serviço é feita utilizando-se o *Web Service Description Language* (WSDL), a qual define uma interface e mecanismos de interação dos serviços além de descrições adicionais como contexto, qualidade do serviço e o relacionamento de serviço para serviços;

d) **publicação e descoberta do serviço**: estas duas camadas utilizam a *UDDI* para descoberta e publicação de informações sobre *Web Services.*

Figura 2: Camadas conceituais dos *Web Services.*



Fonte: KREGER, 2001.

2.3.1 SIMPLE OBJECT ACCESS PROTOCOL (SOAP)

SOAP é um protocolo para troca de informações em um ambiente descentralizado e distribuído, permitindo comunicação entre aplicações (BOX, 2000). Esta comunicação é realizada através de trocas de mensagens, transmitidas através do formato XML, incluindo parâmetros usados na chamada, bem como os dados de resultados. Isto significa que as mensagens podem ser entendidas por quase todas as plataformas de *hardware*, sistemas operacionais, linguagens de programação ou *hardwares* de rede (SEELY, 2002). É um protocolo utilizado para invocar um *Web Service* e também pode ser utilizado para publicar e localizar *Web Services* no registro UDDI.

O SOAP pode, potencialmente, ser utilizado em combinações com uma variedade de outros protocolos, como, HTTP, SMTP e JMS (HENDRICKS, 2002). Suporta *remote procedure call* (RPC), bem como estilo de mensagem assíncrona.

O modelo de dados SOAP oferece definições para tipos de dados mais utilizados como *strings, integer, float, double* e *date.* O processo de traduzir os dados (parâmetros e resultados) em XML é chamado codificação.

Um pacote SOAP consiste de quatro partes:

a) envelope SOAP: define um *framework* que indica o conteúdo da mensagem, quem poderá tratar esta mensagem e se o tratamento é opcional ou obrigatório. É uma estrutura de mensagem SOAP dentro do qual todos os outros elementos sintáticos da mensagem estão encapsulados;
b) codificação SOAP: define mecanismos de serialização que podem ser utilizados para trocar instâncias de tipos de dados definidos na aplicação;
c) RPC SOAP: especifica como embutir *remote procedure call* e respostas dentro das mensagens com o propósito de invocar procedimentos em um sistema remoto;
d) *framework* de Ligação e Transporte SOAP: define um *framework* abstrato para trocar Envelope SOAP entre pares usando um protocolo básico para transporte.

Na especificação do SOAP, outros conceitos importantes são (BOX, 2000):

a) cliente SOAP: é um programa que cria um documento XML contendo a informação necessária para invocar remotamente um método em um sistema distribuído (também pode ser um servidor WEB ou um servidor baseado em aplicações);
b) servidor SOAP: é responsável por executar uma mensagem SOAP e age como um distribuidor e interpretador de documentos;
c) mensagem SOAP: é a forma básica de comunicação entre todos os nodos SOAP.

Uma mensagem SOAP é um envelope contendo cabeçalhos opcionais e um corpo contendo uma mensagem atual com seus parâmetros ou seus resultados. As mensagens SOAP são escritas em XML. Na Figura 3, tem-se um exemplo de um código de uma mensagem SOAP.

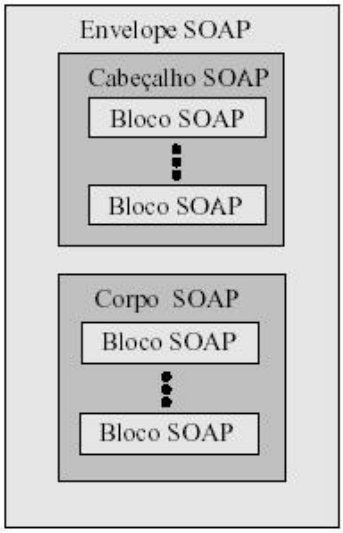
Figura 3: Exemplo de código de envelope SOAP.

```
<env:Envelope xmlns:env="">
  <env:Body>
    <m:getLastTradePrice
      env:encodingStyle="
http://www.w3.org/2001/06/soap-encoding"
      xmlns:m= "
http//www.w3.org/2001/06/quotes"
      <symbol>DIS</symbol>
    </m:GetLastTradePrice>
  </env:Body>
</env:Envelope>
```

Fonte: BOX, 2000.

O envelope SOAP apresenta a estrutura conforme a Figura 4, onde pode-se vizualizar as partes que compõem o envelope. O bloco SOAP é uma construção sintática ou uma estrutura usada para delimitar dados que constitui logicamente uma única unidade computacional. O bloco é identificado por um elemento externo chamado *namespace* URL. O cabeçalho SOAP é uma coleção de zero ou mais blocos, os quais podem ser direcionados para um determinado receptor SOAP dentro do caminho da mensagem. O corpo SOAP é uma coleção de zero ou mais blocos direcionados para o último receptor SOAP.

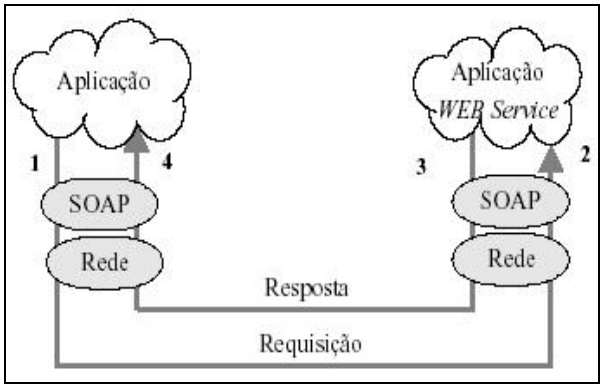
Figura 4: Estrutura do envelope SOAP



Fonte: BOX, 2000.

A chamada do serviço utilizando SOAP ocorre conforme apresentado na Figura 5. A aplicação (1) requisita uma mensagem SOAP e invoca a operação do serviço através de um provedor de *Web Service*. O **Solicitante de Serviço** apresenta a mensagem junto com o endereço de rede do provedor de *Web Service.* A infra-estrutura da rede (2) entrega a mensagem para um servidor SOAP. O Servidor SOAP redireciona a mensagem requisitada para o **Provedor de Serviço** *Web Service.* O servidor WEB (3) é responsável por processar uma mensagem de requisição e formular uma resposta. A mensagem (4) é redirecionada através da infraestrutura SOAP. Quando a mensagem XML chega no nodo requisitante, é convertida para a linguagem de programação, sendo então entregue para a aplicação.

Figura 5: Mensagens XML usando SOAP



Fonte: KREGER, 2001.

Para que um serviço seja utilizado faz-se necessário que o cliente saiba localizá-lo. Esta localização pode ser feita através da UDDI.

2.3.2 WEB SERVICE DESCRIPTION LANGUAGE (WSDL)

A WSDL é uma linguagem que utiliza XML para servir como um meio para fornecedores de serviços disponibilizarem suas interfaces de acesso. Estas interfaces, representadas em XML, contêm informações sobre todas as funcionalidades do serviço bem como os tipos de dados necessários para a sua utilização, ou seja, tudo que uma aplicação cliente precisa saber para poder usar um serviço. A representação dos dados ocorre obedecendo a um dos objetivos dos *Web Services* que é a transparência da implementação. Assim esta forma de representação não está ligada a nenhuma linguagem especifica de programação. WSDL foi definida então como sendo a linguagem padrão para descrever a interface e localização de um serviço.

Especificamente WSDL fornece um número de peças-chave de informação:

a) a definição do formato das mensagens que são passadas entre dois pontos finais usando seus elementos < *types* > e < *message* > e definições apropriadas de esquemas (*schema*);
b) a semântica do serviço: como ele deve ser chamado para fazer uma transmissão síncrona de solicitação e resposta, síncrona apenas de resposta ou comunicação assíncrona;
c) o ponto final e o transporte do serviço através do elemento < *service* >: isto é, quem fornece o serviço;
d) uma ligação através do elemento < *binding* >: isto é, como o serviço é alcançado.

Na Figura 6, é possível observar como um tipo de dados é representado na linguagem WSDL.

Figura 6: Exemplo de código WSDL.

```
- <types>
  - <xs:schema targetNamespace="urn:TesteIntf" xmlns="urn:TesteIntf">
    - <xs:simpleType name="TEnumTest">
      + <xs:restriction base="xs:string">
      </xs:simpleType>
    - <xs:complexType name="TDoubleArray">
      - <xs:complexContent>
        - <xs:restriction base="soapenc:Array">
            <xs:sequence />
            <xs:attribute ref="soapenc:arrayType" n1:arrayType="xs:double[]"
              xmlns:n1="http://schemas.xmlsoap.org/wsdl/" />
          </xs:restriction>
        </xs:complexContent>
      </xs:complexType>
    - <xs:complexType name="TMyEmployee">
      - <xs:sequence>
          <xs:element name="LastName" type="xs:string" />
          <xs:element name="FirstName" type="xs:string" />
          <xs:element name="Salary" type="xs:double" />
        </xs:sequence>
      </xs:complexType>
    </xs:schema>
  </types>
```

2.3.3 UNIVERSAL DISCOVERY AND DESCRIPTION INTEGRATION (UDDI)

"A especificação UDDI define um número de serviços de busca destinados a permitir que as aplicações clientes encontrarem e obterem informações para acessar um *Web Service*". (SAGANICH, 2001).

UDDI atualmente prove três serviços específicos:

a) *Traditional White Pages* para procurar um *Web Service* pelo nome;

b) *Traditional Yellow Pages* para procurar um *Web Service* pelo tópico;

c) *Green Pages* para buscas mais genéricas baseadas nas características de um *Web Service.*

Fornecedores de serviços UDDI tipicamente operam um serviço conhecido como UDDI *Business Registry,* ou UBR, que pode ser acessado para publicar e solicitar informações sobre um dado *Web Service.*

2.4 WEB SERVICES COM DELPHI 7

O que torna possível a comunicação do Delphi com os *Web Services* é o *Simple Object Access Protocol* (SOAP).

A ferramenta Delphi possui um mapeamento bidirecional entre WSDL e interfaces. Isto significa que é possível ter um arquivo WSDL e gerar uma interface para ele. Sendo assim, é possível criar uma aplicação cliente com requisições SOAP através destas interfaces e usar um componente especial do Delphi que permita converter suas interfaces de requisições locais em chamadas SOAP (CANTÚ, 2001a).

Ao criar aplicações w*eb services* utilizando o Delphi, alguns componentes fundamentais para a implementação do servidor são apresentados:

a) **tHTTPSoapDispatcher:** atua como despachante recebendo as mensagens entrantes e as encaminhando para o objeto especificado em sua propriedade *Dispatcher* para que sejam decodificadas;
b) **tHTTPSoapPascalInvoker:** este componente recebe a mensagem vinda do THTTPSoapDispatcher e a interpreta disparando o método correspondente à solicitação. Ele ainda codifica o retorno do método para o padrão SOAO/XML;
c) **tWSDLHTMLPublish:** é responsável por publicar todas as informações registradas pelo *Web Service* em WSDL que faz com que qualquer pessoa possa adquirí-las para implementar o cliente deste serviço, mesmo que seja em outra ferramenta que não o Delphi.

Para publicar e disponibilizar uma interface implementada no servidor usando o Delphi é necessário que a mesma seja derivada da classe IInvokable.

Interfaces invocáveis são interfaces que são compiladas com a informação de *Runtime Type Information* (RTTI). No servidor esta RTTI interpreta chamadas de métodos vindas de aplicações clientes para que elas possam ser corretamente executadas e no cliente, a RTTI é usada para gerar dinamicamente uma tabela de métodos para fazer

chamadas para os métodos das interfaces. Um exemplo desta interface pode ser observada na Figura 7.

Figura 7: Exemplo de uma interface invocável.

```
IEncodeDecode = interface(IInvokable)
  ['{C527B88F-3F8E-1134-80e0-01A04F57B270}']
    function EncodeValue(Value: Integer): Double; stdcall;
    function DecodeValue(Value: Double): Integer; stdcall;
end;
```

Os três passos fundamentais para criar aplicações clientes que acessem os *Web Services* no Delphi são:

a) importar as definições de um WSDL documento para que a aplicação possa definir e registrar as interfaces invocáveis e os tipos que estão incluídos na aplicação *Web Service*. Para importar estas informações o Delphi disponibiliza uma ferramenta chamada *WSDL Importer*;
b) obter uma interface invocável e chamá-la para invocar o *Web Service*;

processar os cabeçalhos das mensagens SOAP que passam entre o cliente e o servidor.

3 DESENVOLVIMENTO

O protótipo tem como objetivo possibilitar a troca de informações entre agentes de viagens e seus fornecedores utilizando a tecnologia *Web Services*. Ele consiste em três aplicações: aplicação para o fornecedor (aplicação fornecedora), *Web Service* e aplicação do agente de viagem (aplicação cliente).

3.1 REQUISITOS

O turismo é considerado, atualmente, como a área da economia que mais cresce no mundo. Para acompanhar este crescimento, faz se necessário que as agências de viagens tenham cada vez mais acesso a informações de seus fornecedores, possibilitando assim um melhor atendimento aos seus clientes. Atualmente a melhor forma de obter estas informações de seus fornecedores está por meio da internet.

No Brasil, a internet já está presente no setor de viagens, em mais de trinta e cinco mil hotéis, cinqüenta locadoras de veículos, quatrocentas empresas aéreas, trinta mil agências de viagens de turismo e 3,5 milhões de consumidores. Este setor movimenta cifras que crescem 20% por mês. Esses números, é claro, são apenas estimados. Ninguém sabe ao certo que mercado consumidor a internet esconde (TOMELIN, 2001).

Conforme Montaner (2001), “A informática aplicada ao setor turístico – turismática – trata de concepção, realização e utilização dos sistemas que processam informação relacionada com a gestão – reservas, vendas, contabilidade, bando de dados, etc. – do tratamento dos dados encaminhados para prestar os diferentes serviços turísticos”. Assim, o principal objetivo das agências viagens é poder atender seus clientes da melhor maneira possível, ou seja, atendê-lo de forma eficiente e eficaz.

Neste sentido, este sistema deve proporcionar que fornecedores hoteleiros possam disponibilizar seus recursos on-line e que estes recursos possam ser pesquisados e alocados pelos agentes de viagens. Desta maneira os agentes de viagens poderão ter certeza que os recursos pesquisados estarão disponíveis, sem ter a necessidade de realizar a alocação destes recursos dias antes do desejado. A idéia é que cada agente de viagem tenha o maior

número possível de fornecedores cadastrados que possua esse sistema, pois, assim, a pesquisa de recursos cada vez mais trará resultados que satisfaçam seus clientes.

O fornecedor poderá configurar o acesso da aplicação à sua base indicando alguns dados, como por exemplo, qual banco de dados sua aplicação utiliza e os outros dados particulares ao banco que for escolhido. Assim, será possível que a aplicação fornecedora possa acessar a base de dados e permita que o fornecedor indique a localização das informações que deseja disponibilizar para os agentes de viagens.

Ainda nesta aplicação o fornecedor pode verificar todas as requisições feitas para seus recursos e retorná-las informando se esta foi aceita ou não, e caso não foi aceita, será possível retornar ao agente de viagens qual o motivo da não aceitação.

O sistema resolverá um dos maiores problemas das agências de viagens, ou seja, com ele, não será necessário ter que acessar páginas de internet atrás de dos serviços de todos os seus fornecedores, os dados estarão disponíveis pelo sistema através das consultas que poderão ser realizadas.

3.1.1 APLICAÇÃO FORNECEDORA

A aplicação fornecedora é responsável por parametrizar quais os dados que os agentes de viagens terão acesso através de consultas. A funcionalidade principal desta aplicação é interfacear a base de dados do fornecedor, ou seja, permitir que os *Web Services* possam identificar a localização dos dados pertinentes às consultas. Desta forma o agente de viagens poderá, em outro momento, identificar facilmente quais os dados relevantes para a sua consulta.

Para que todos os fornecedores possam disponibilizar as mesmas informações, facilitando assim as consultas dos agentes de viagens, serão pré-definidas quais as informações possíveis de serem parametrizadas, como por exemplo: número do quarto, classificação, cabendo ao fornecedor apenas informar a localização destas informações em seu sistema.

3.1.2 APLICAÇÃO WEB SERVICE

A aplicação *Web Service* é a responsável pela interação entre os agentes de viagens e seus fornecedores. Nela devem ser implementados métodos que permita ao agente de viagens ter acesso às informações do fornecedor.

Além disso, ela deve possibilitar que os agentes de viagens filtrem as informações disponibilizadas pelos fornecedores, acessando, desta forma, somente as informações pertinentes. Para fazer o filtro, serão disponibilizados os campos que foram pré-definidos e anteriormente liberados para os fornecedores configurarem suas localizações. Nestes campos os agentes de viagens poderão indicar quais valores desejam encontrar.

Também é possível que depois de ter selecionado a melhor opção resultante das pesquisas, o agente de viagens possa enviar uma solicitação de requisição de recursos, para que posteriormente o fornecedor verifique estas requisições e as avalie, autorizando-as ou caso contrário informando o motivo da não autorização.

3.1.3 APLICAÇÃO CLIENTE

A aplicação cliente acessa os *Web Services* e utiliza seus métodos para visualizar os dados liberados pelo fornecedor, garantindo que os dados que estão sendo pesquisados reflitam realmente a situação atual daquele fornecedor.

Esta aplicação será responsável por aglutinar as informações resultantes de pesquisas feitas em um ou mais fornecedores e disponibilizá-las para os agentes de viagens de uma forma legível, permitindo que o mesmo possa identificar os recursos e escolher entre eles qual a opção que mais se adapta à sua necessidade.

Além disso, permitirá que o agente de viagens envie uma mensagem para o *Web Service* requisitando a alocação de algum recurso.

3.2 ESPECIFICAÇÃO

Para a especificação do sistema foram utilizados o Diagrama de Entidade e Relacionamento (DER) e o Diagrama de Fluxo de Dados (DFD). Para a criação destes diagramas foi utilizada a ferramenta Power Designer 6.1.

3.2.1 LISTA DE EVENTOS

A lista de eventos do sistema está descrita abaixo:

a) fornecedor configura o acesso aos dados de sua base na aplicação fornecedora;

b) fornecedor configura campos que deseja liberar para consultas;

c) agente de viagens cadastra os fornecedores;

d) agente de viagens cadastra clientes;

e) agente de viagens cadastra filtros para utilizar nas pesquisas;

f) agente de viagens pesquisa recursos dos fornecedores;

g) agente de viagens envia requisição de alocação de recurso para o fornecedor;

h) fornecedor avalia requisição enviada pelos agentes de viagens;

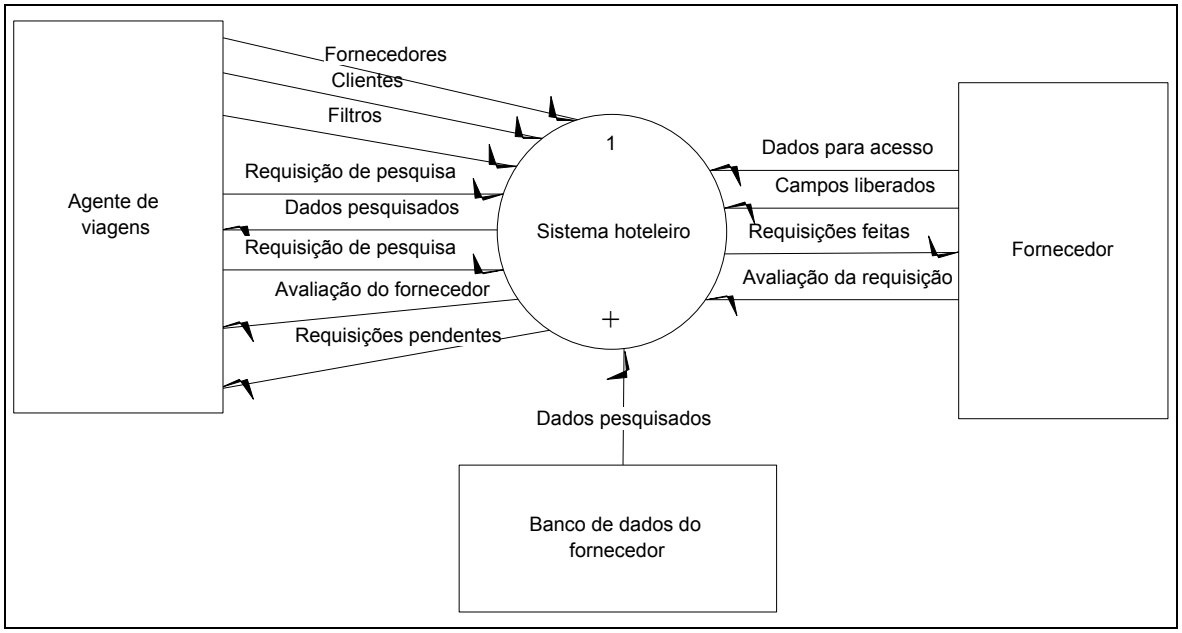
i) agente de viagem consulta avaliação do fornecedor;

j) agente de viagem consulta requisições pendentes.

3.2.2 DIAGRAMA DE CONTEXTO

O Diagrama de Contexto do sistema é apresentado na Figura 8.

Figura 8: Diagrama de contexto



3.2.3 DIAGRAMAS DE FLUXO DE DADOS

Os Diagramas de Fluxo de Dados da aplicação estão apresentados nas figuras 9 e 10. Estes diagramas estão segmentados por entidade.

Figura 9: Diagrama dos eventos do fornecedor.

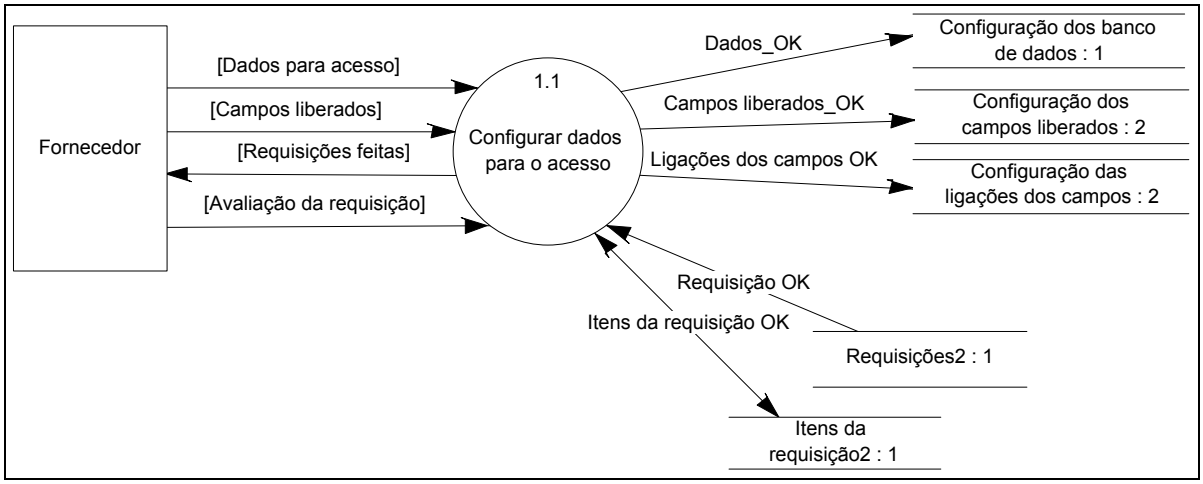
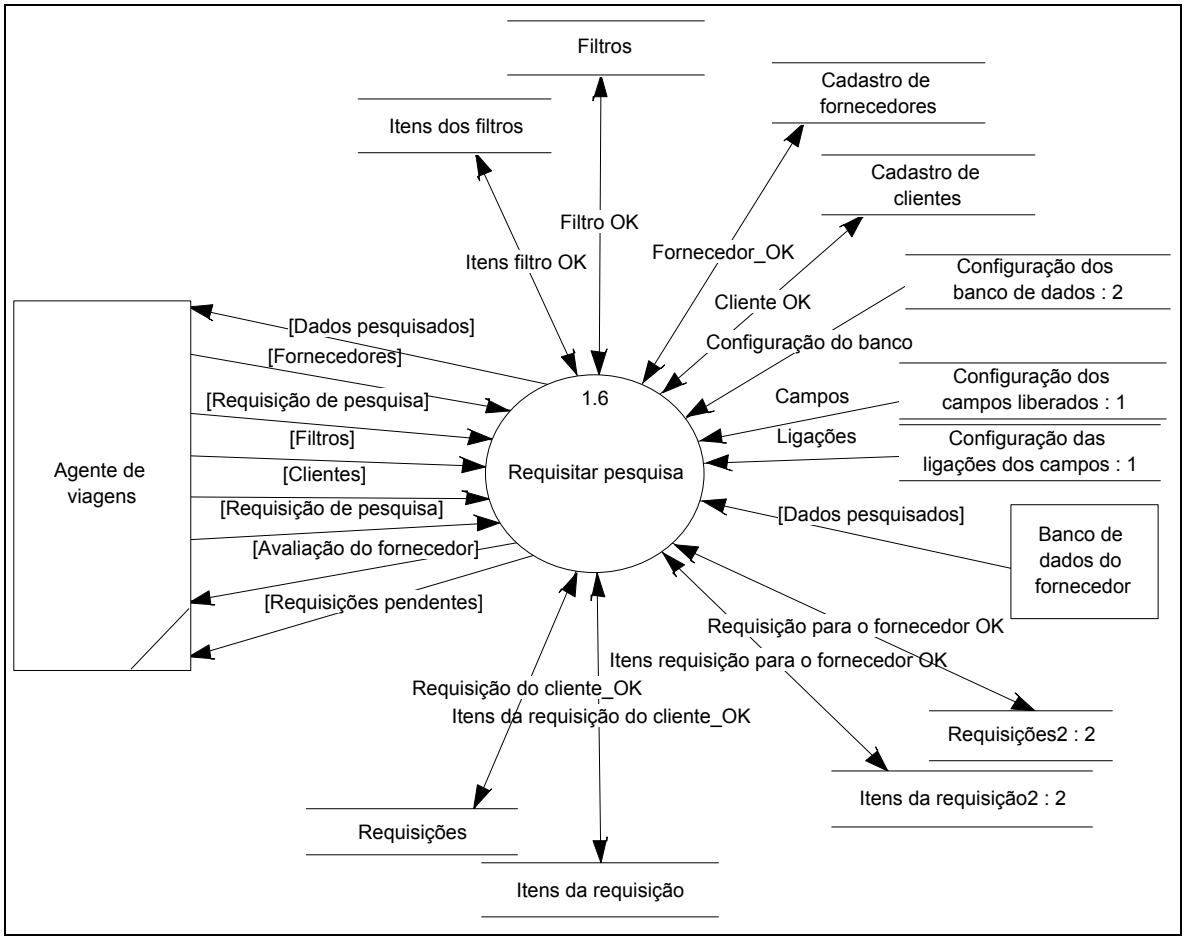


Figura 10: Diagrama dos eventos dos agentes de viagens.



3.2.4 APLICAÇÃO FORNECEDORA

A aplicação do fornecedor armazena as informações de acesso ao banco, a localização dos dados que foram liberados e as requisições. Seu diagrama de entidade e relacionamento (DER) pode ser visto na Figura 11 e Figura 12. Foi utilizado para o armazenamento das informações da aplicação fornecedora o banco de dados Interbase.

Figura 11: DER lógico da aplicação fornecedora

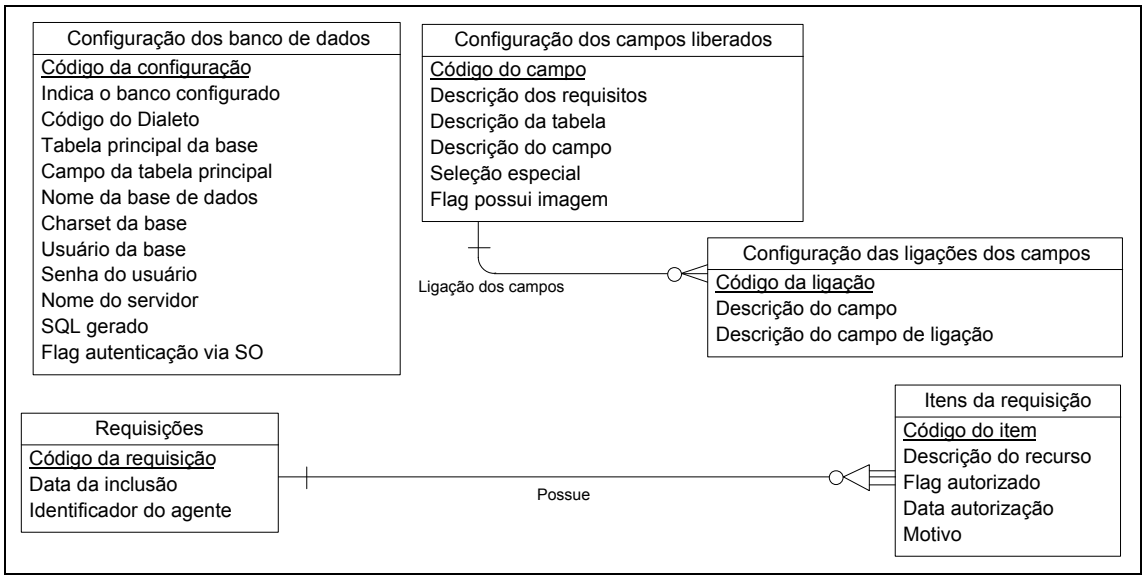
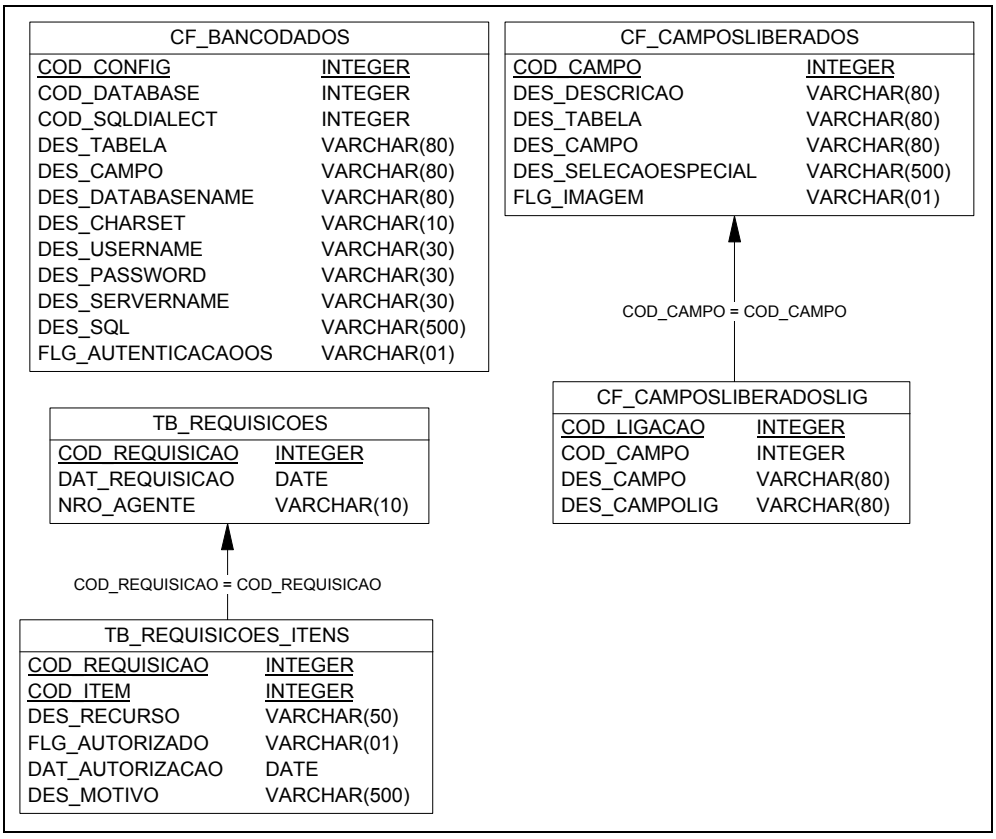


Figura 12: DER físico da aplicação fornecedora



O dicionário de dados pode ser visualizado no Apêndice 1.

3.2.5 APLICAÇÃO CLIENTE

Esta aplicação armazena os dados referentes ao cadastro dos fornecedores e a localização dos seus *Web Services,* além de guardar os pedidos de alocação de recursos disponíveis nos fornecedores que foram previamente cadastrados e também armazenar s dados referentes aos filtros feitos para facilitar uma nova consulta que possa ser feita. As Figura 13 e Figura 14 apresentam o DER desta aplicação. Foi utilizado para armazenamento dos dados da aplicação cliente o banco de dados Interbase.

Figura 13: DER lógico da aplicação cliente.

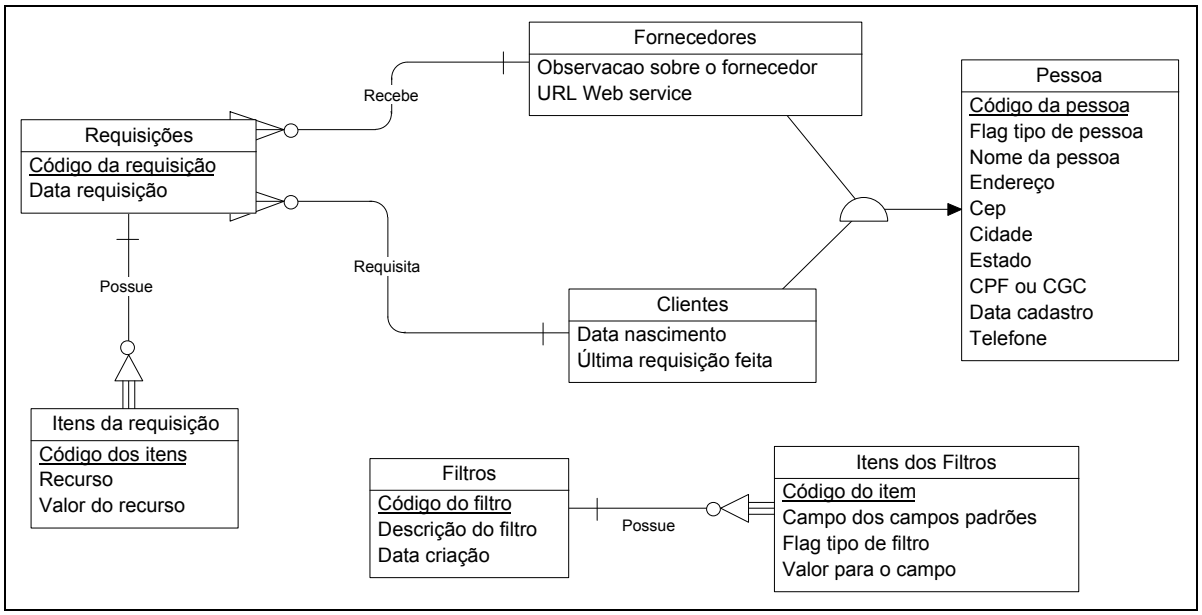
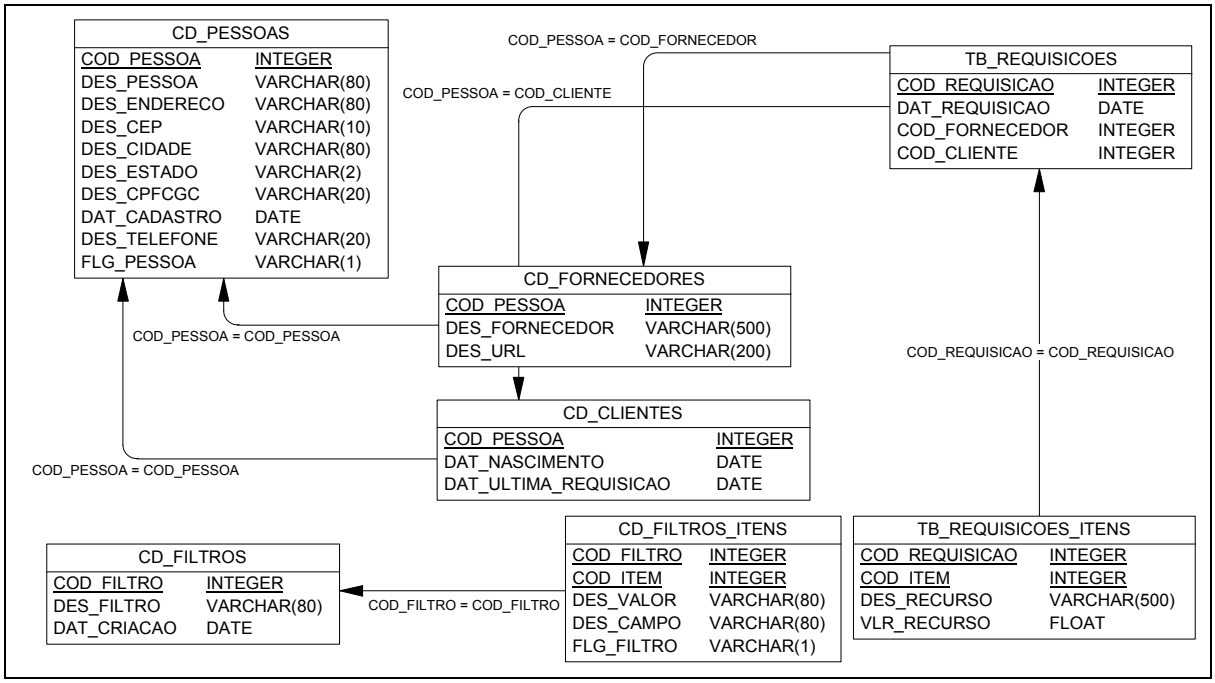


Figura 14: DER físico da aplicação cliente.

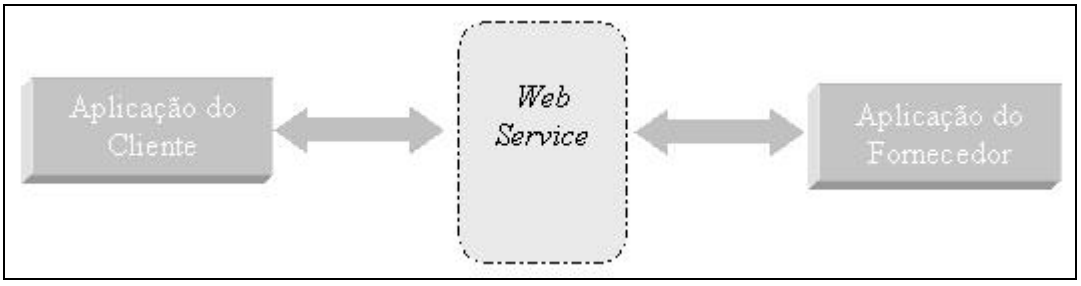


O dicionário de dados pode ser visualizado no Apêndice 2.

3.3 IMPLEMENTAÇÃO

A Figura 15 demonstra a estrutura básica do funcionamento do protótipo, onde os *Web Services* serão intermediários entre a comunicação da aplicação cliente com a aplicação fornecedora. Desta forma a programação da aplicação cliente fica flexível, podendo ser desenvolvida em qualquer plataforma e ou linguagem de programação.

Figura 15: Arquitetura do sistema



A implementação do sistema foi feita utilizando o ambiente de desenvolvimento Delphi 7 com os bancos de dados Interbase e MS-SQL Server.

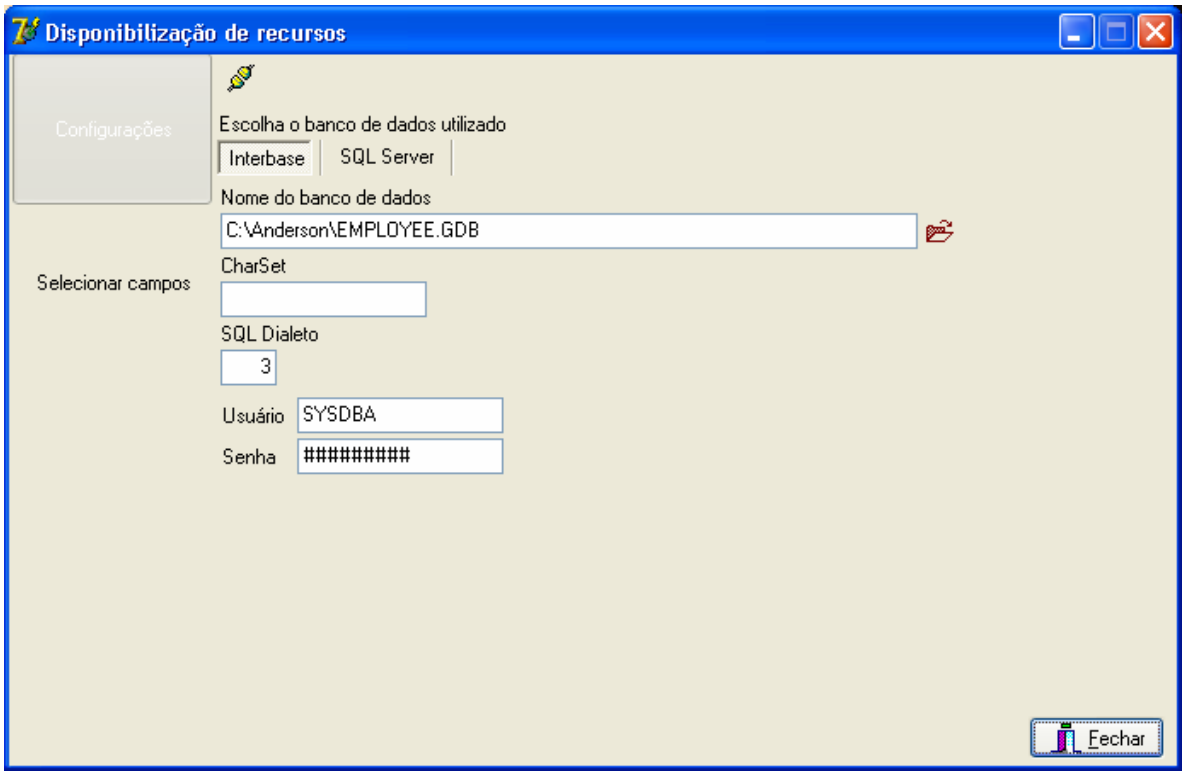
Para acessar os bancos de dados pelo Delphi, foram utilizados os componentes Dbexpress, visto que eles permitem o acesso a vários bancos de dados.

A implementação foi feita em três aplicações distintas. A seguir serão demonstrados os principais aspectos de implementação.

3.3.1 APLICAÇÃO FORNECEDORA

Nesta aplicação é necessário que o fornecedor configure o acesso do sistema ao seu banco de dados. Este procedimento poderá ser realizado informando os dados pertinentes ao banco selecionado, como pode ser verificado na Figura 16.

Figura 16: Tela de configuração de acesso ao banco de dados dos fornecedor.

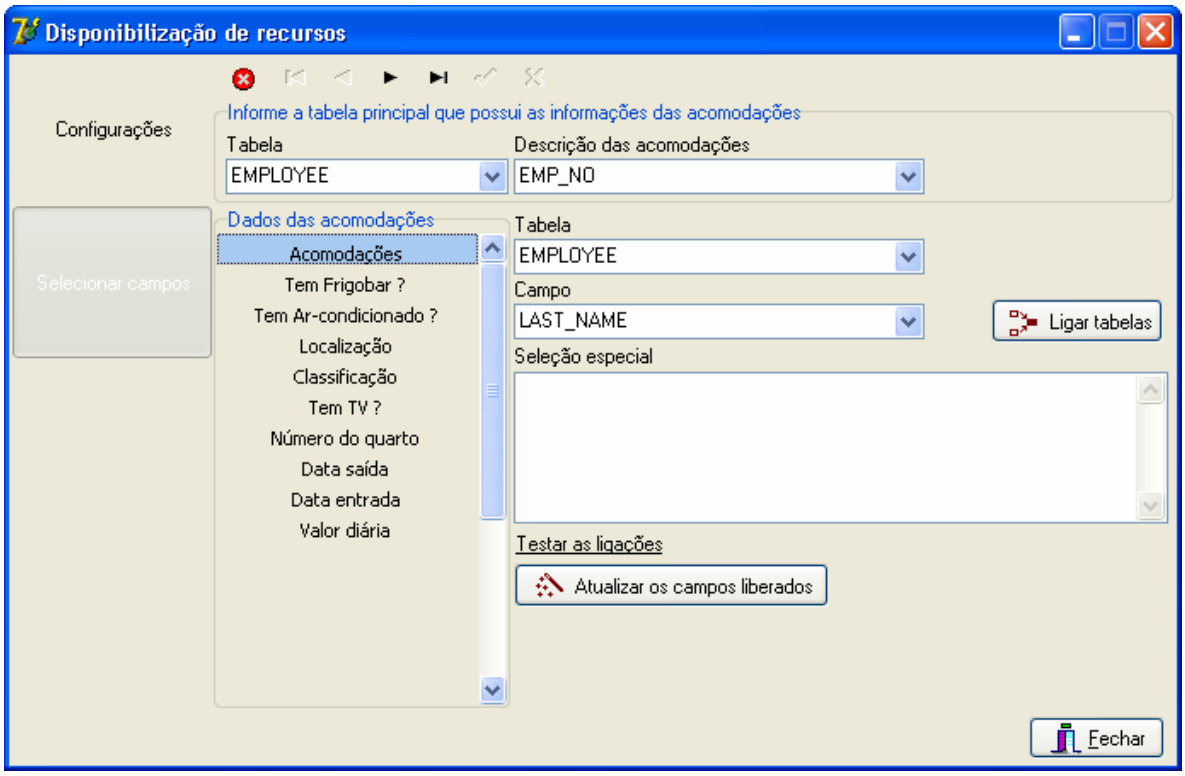


Além de configurar o acesso ao seu banco de dados, o fornecedor precisa configurar os dados que são disponibilizados para os agentes de viagens. Os dados a serem disponibilizados são padronizados sendo que, o que muda de um fornecedor para o outro é a localização das informações dentro do sistema.

Na Figura 17 pode ser visto a tela de configuração destes dados, onde se encontram os campos já pré-definidos e é possível parametrizar a localização dos mesmos, indicando qual tabela e qual campo contém a informação.

Antes de configurar os campos padrões é necessário eleger qual é a tabela principal do sistema, ou seja, a tabela que possui as informações das acomodações.

Figura 17: Tela de configuração dos campos pré-determinados.

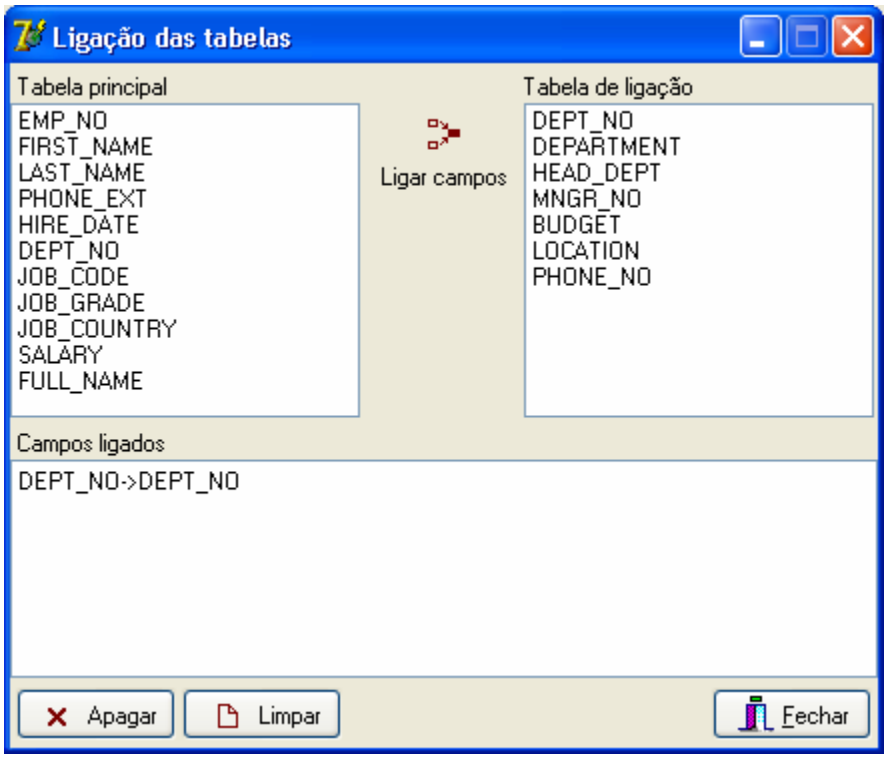


Ao escolher uma tabela como sendo a tabela principal, poderão ser configurados os outros campos, que deverão possuir alguma ligação com a tabela principal. Esta ligação é parametrizada na tela indicada na Figura 18.

A ligação funcionará como o *join* feito em um SQL, por isso a configuração correta é muito importante, para evitar que os dados se multipliquem desnecessariamente.

Após a configuração dos campos é possível verificar o resultado das ligações, para averiguar se elas foram corretamente indicadas e atualizar os campos liberados.

Figura 18: Configuração das ligações das tabelas com a tabela principal.



3.3.2 APLICAÇÃO *WEB SERVICES*

Para a programação do *Web Service* foi necessário instalar um componente do Windows chamado *Internet Information Service* (IIS). O IIS é um conjunto de ferramentas voltadas para o FTP e WEB. Após instala-lo é possível testar os *Web Services* localmente, pois a máquina se torna um servidor de páginas WEB.

A estrutura básica do *Web Service* foi criada pelo *wizard* do Delphi, que cria automaticamente três *units*:

a) *unit* que contém o *SOAP Webmodule*, onde encontram-se os componentes THTTPSoapDispatcher, THTTPSoapPascalInvoker, TWSDLHTMLPublish, que, como já comentado anteriormente, são fundamentais para a implementação e publicação do serviço;

b) *unit* que contém uma Interface *Invoke* para o Web Service, onde estão definidos as interfaces que poderão ser invocadas no *Web Service* e também os tipos de dados usados por estas interfaces, como pode ser observado no Quadro 1;

Quadro 1: Exemplo da implementação da interface *Invoke*.

```
type
  TEnumTest = (etNone, etAFew, etSome, etAlot);
  TDoubleArray = array of Double;
  TMyEmployee = class(TRemotable)
  private
    FLastName: AnsiString;
    FFirstName: AnsiString;
    FSalary: Double;
  published
    property LastName: AnsiString read FLastName write FLastName;
    property FirstName: AnsiString read FFirstName write FFirstName;
    property Salary: Double read FSalary write FSalary;
  end;
  { Invokable interfaces must derive from IInvokable }
  ITeste = interface(IInvokable)
  ['{F04CC530-1722-4321-8BA9-219623376F55}']
    { Methods of Invokable interface must not use the default }
    { calling convention; stdcall is recommended }
    function echoEnum(const Value: TEnumTest): TEnumTest; stdcall;
    function echoDoubleArray(const Value: TDoubleArray): TDoubleArray; stdcall;
    function echoMyEmployee(const Value: TMyEmployee): TMyEmployee; stdcall;
    function echoDouble(const Value: Double): Double; stdcall;
  end;
```

c) *unit* que contém a implementação da interface *invoke* definida anteriormente.

O *Web Service* pode disponibilizar os dados do fornecedor através de um *SOAP Server Data Module,* assim denominado no Delphi. Desta forma é publicado um *DataSetProvider*, que permite às aplicações clientes mostrar e alterar estas informações recebidas através de pacotes SOAP que são transmitidos para o cliente conforme forem solicitados. Esta forma de disponibilizar os dados não se mostrou eficiente, levando muito tempo para retornar os dados pesquisados. Levando em consideração que os dados vindos do fornecedor não sofrerão nenhuma alteração, foi criado um método na aplicação *Web Service* que retorna um objeto composto por: uma lista de colunas e outra dos dados destas colunas, ou seja, através deste objeto é possível ter acesso aos dados resultantes da pesquisa realizada. Esta forma mostrou-se muito mais eficaz que a anterior, retornando os valores mais rapidamente, além de permitir maior flexibilidade no envio de filtros para as pesquisas. A estrutura deste objeto pode ser vista no Quadro 2.

Quadro 2: Declaração da classe que será retornada na pesquisa dos dados.

```
TSoapDataPacket = class(TRemotable)
private
  FColDescArray   : TColDescArray;
  FIndexDescArray : TIndexDescArray;
  FRowArray       : TRowArray;
  FTableName      : string;
public
published
  property ColDescArray: TColDescArray
           read FColDescArray write FColDescArray;
  property IndexDescArray: TIndexDescArray
           read FIndexDescArray write FIndexDescArray;
  property RowArray: TRowArray
           read FRowArray write FRowArray;
  property TableName: string
           read FTableName write FTableName;
end;

IWSService = interface(IInvokable)
['{0A14EB8D-4CCA-4F3B-97C8-7BEEF3066418}']
  function Filtrar(SQL: String;
                   var SoapDataPacket: TSoapDataPacket): String; stdcall;
end;
```

Os principais eventos criados na aplicação *Web service* são:

a) *wsEnviarRequisicao:* este evento envia a requisição de um recurso escolhido pelo agente de viagem para o fornecedor. A requisição é armazenada numa das tabelas do sistema do fornecedor, para que o fornecedor possa visualizá-la e indicar se ela foi aceita ou não. Após a indicação do fornecedor é enviada uma mensagem para o agente de viagem que a solicitou informando sua situação;

b) *wsVerificarRequisicao:* o agente de viagem utilizará este recurso para pesquisar o estado de sua reserva, verificando em qual das situações ela se encontra: aguardando verificação, aceita, recusada;

c) *wsCamposPadroes:* este evento retorna uma lista com os campos padrões, ou seja os campos pré-definidos para serem configurados.

O uso do *Web Service* facilita a reutilização de código, ou seja, é possível acessá-lo de qualquer plataforma ou linguagem de programação, sem a necessidade de alterar o código do *Web Service.* Por isso, a aplicação do cliente preocupa-se apenas em acessar os

recursos fornecidos pelo *Web Service* e fazer com que os dados sejam exibidos de uma forma legível para o fornecedor.

3.3.3 APLICAÇÃO CLIENTE

O ponto principal da aplicação cliente é a utilização das funcionalidades implementadas nos *Web Services*, para ter acesso aos dados de seus fornecedores. Para utilizar estas funcionalidades é necessário importar o documento WSDL da aplicação *Web Service.*

Para importar este documento foi utilizada a ferramenta do Delphi chamada *WSDL Importer* que necessita apenas a localização do serviço para poder importá-lo. Ao importar o documento, o *WSDL Importer* gera uma *unit* com todas as interfaces e definições de classes que são necessárias para a chamada do serviço. O Apêndice 3 mostra a *unit* gerada a partir de um documento WSDL importado pelo Delphi.

Para utilizar os métodos das classes existentes no documento WSDL importado é necessário primeiramente adicionar a *unit* gerada pelo Delphi à aplicação. A Figura 19 e Figura 20 mostra a utilização desses métodos. Para sua utilização foi criado um objeto com o mesmo tipo definido na classe e uma função que instancia o objeto caso o mesmo ainda não tenha sido criado ou retorna-o caso contrário.

Figura 19: Declaração do objeto que utiliza a classe existente no documento importado.

```
private
  FWSService : IWSServicel;
public
  FormAtivo : TForm;
  function  GetWSService: IWSServicel;
  procedure LiberaForms;
end;
```

Figura 20: Utilização de métodos descritos no documento WSDL.

```
function TFPrincipal.GetWSService: IWSServicel;
begin
  if FWSService = nil then begin
    FWSService := GetIWSServicel(False,
                  CadFornecedores.FieldByName('DES_URL').AsString);
    (FWSService as IRIOAccess).RIO.OnBeforeExecute := HTTPRIO1BeforeExecute;
    (FWSService as IRIOAccess).RIO.OnAfterExecute  := HTTPRIO1AfterExecute;
  end;
  Result := FWSService;
end;
```

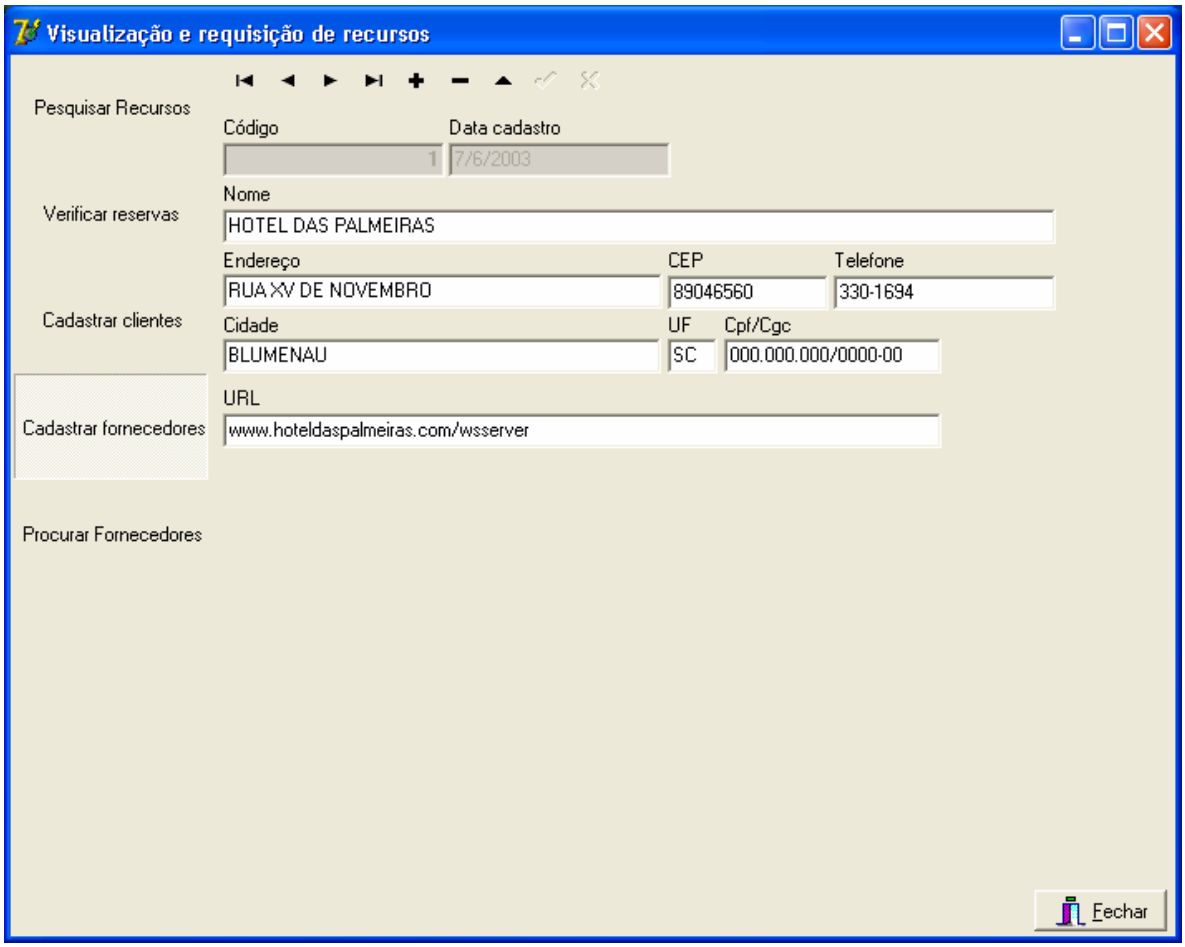
O objeto HTTPRIO1 [1] permite visualizar a troca de mensagens SOAP que acontece a cada requisição de um serviço do *Web Service.*

Para que o agente de viagem possa procurar por recursos de seus fornecedores foram criadas algumas funcionalidades em sua aplicação. Foi implementado um cadastro simples de fornecedores e clientes, identificado na Figura 21.

Ao cadastrar seus fornecedores o agente de viagem deve preencher no campo URL a localização do *Web Service*, para que a aplicação o considere ao fazer a pesquisa por recursos. Ao realizar a pesquisa por recursos disponíveis o sistema possibilita que os agentes de viagem escolham em quais fornecedores serão verificados os recursos disponíveis. Selecionado os fornecedores, o sistema irá tentar se conectar aos *Web Services* informados no cadastro destes fornecedores. Caso o *Web Service* informado não esteja disponível, o mesmo será descartado da consulta, caso contrário será apresentada uma lista com seus recursos que se enquadram no filtro.

[1] THTT PRIO é utilizado para gerar chamadas estáticas para invocar interfaces em uma aplicação remota *Web Service.* Ele executa os métodos existentes no documento WSDL indicado, codificando a chamada do método como uma requisição SOAP e enviando uma requisição HTTP para a aplicação *Web* Service. Ele desmonta a mensagem HTTP de resposta para obter os valores resultantes.

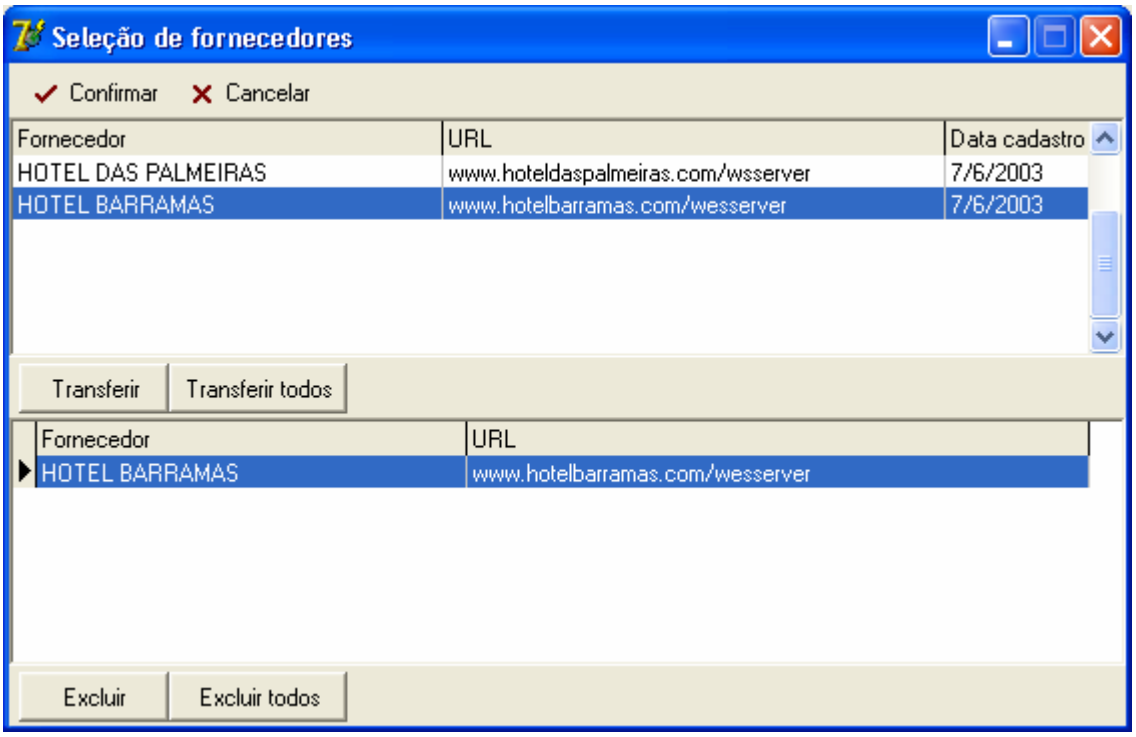
Figura 21: Cadastro de fornecedores.



Para selecionar em quais fornecedores deverá ser feita a pesquisa de recursos, foi implementada a rotina demonstrada na Figura 22 que permite ao agente de viagens selecionar um ou mais fornecedores que serão pesquisados, evitando assim que seja feita pesquisas em fornecedores desnecessariamente.

Como padrão, inicialmente são definidos para a pesquisa todos os fornecedores que possuírem o endereço do *Web Service* informado em seu cadastro, podendo o agente de viagens selecionar apenas aqueles que eles desejam visualizar os recursos.

Figura 22: Selecionar fornecedores para filtrar recursos.



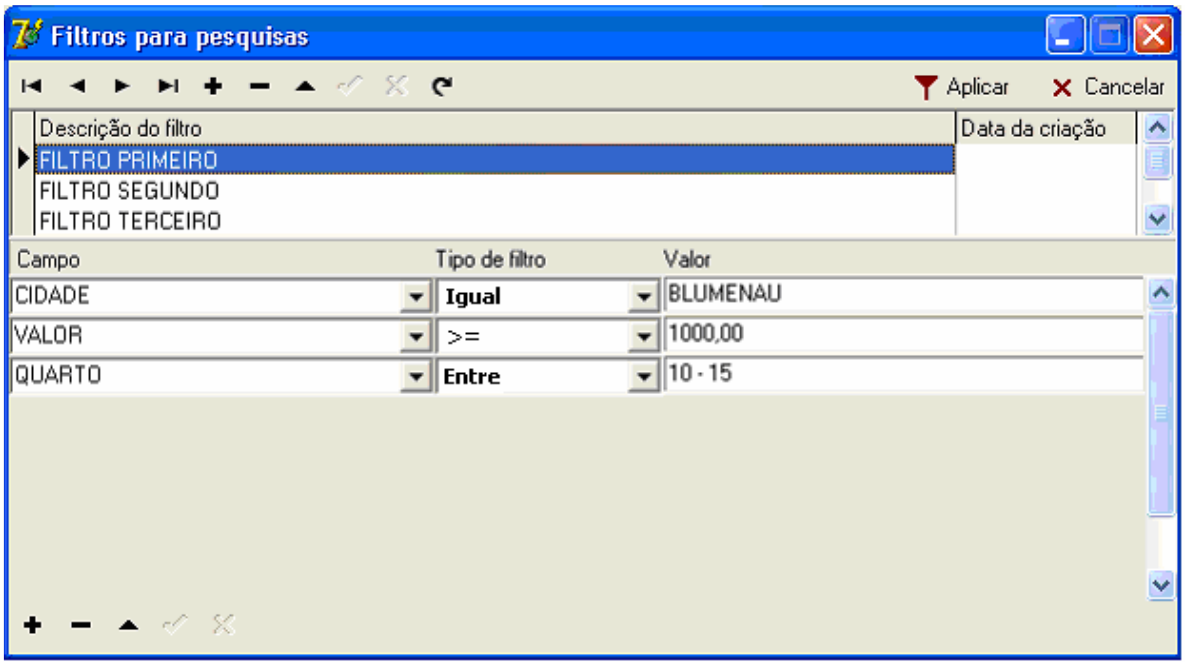
Também para facilitar as pesquisas por recursos, foi desenvolvido um filtro, para que apenas os dados pertinentes apareçam nos resultados das pesquisas.

O filtro foi implementado da seguinte maneira: o agente de viagem poderá configurar quais os valores que ele deseja encontrar em determinados campos. O filtro será feito a partir dos campos padrões, ou seja, assim o agente de viagem pode ter certeza que a pesquisa feita em um fornecedor, poderá ser feita em qualquer outro.

Os filtros que foram criados serão gravados para que posteriormente possam vir a ser reutilizados sem a necessidade do agente de viagens ter que digitá-los novamente.

A tela de criação e configuração dos filtros pode ser visualizada na Figura 23.

Figura 23: Filtro para a realização das pesquisas



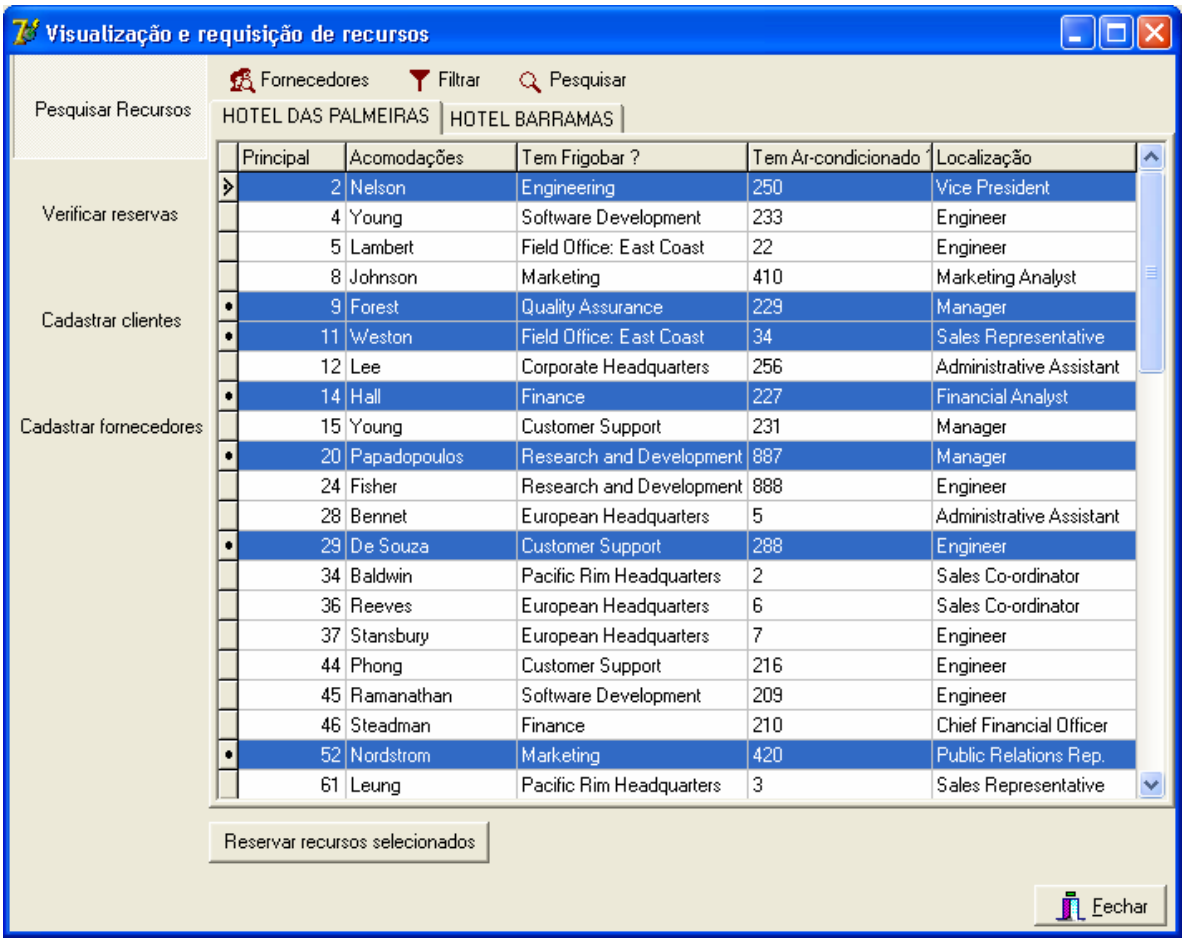
A aplicação cliente solicita aos *Web Services* a pesquisa dos recursos e os *Web Services* acessam a base dos fornecedores, e procuram nos campos parametrizados as informações que foram solicitadas, retornando os dados que foram encontrados.

No processo de pesquisa de recursos pode ser apontada uma vantagem no uso dos *Web Services*: é indiferente para a aplicação cliente o banco de dados que o fornecedor está utilizando, pois qualquer um que seja o *Web Servic*ele irá responder da mesma forma para a aplicação cliente.

Com as respostas das requisições, a aplicação monta uma tabela para que os agentes de viagens possam visualizá-la e escolher qual recurso deseja fazer uma requisição.

A tela da pesquisa pode ser visualizada na Figura 24.

Figura 24: Pesquisa de recursos.



3.4 RESULTADOS E DISCUSSÃO

O sistema desenvolvido apresenta algumas limitações que podem ser facilmente contornadas, mas como o objetivo principal do trabalho visava a utilização dos *Web Services* no desenvolvimento de aplicações corporativas não foram levadas em consideração.

Algumas destas limitações observadas e que podem ser citadas são:

a) A aplicação cliente poderia ser desenvolvida com ferramentas para acesso via Web, possibilitando assim o acesso direto dos clientes ao sistema, sem ter que recorrer aos agentes de viagens para conseguir reservar algum recurso;

b) A configuração dos campos na aplicação fornecedora ficou um pouco rígida, ou seja, não ficou muito fácil de ser feita, pois o ideal seria que os fornecedores

utilizassem um sistema para gerenciar seus recursos desenvolvidos pelo mesmo desenvolvedor da aplicação *Web Services*, ou que a aplicação que interfaceia a base do fornecedor verificasse todas as ligações existentes em suas tabelas.

A primeira dificuldade encontrada no desenvolvimento das aplicações foi no momento que se percebeu a necessidade de interagir com as bases de dados dos diversos fornecedores que poderão utilizar o sistema, sem ter que se preocupar com sua estrutura e a localização interna dos dados. A solução para este problema foi a criação de uma aplicação que "interfaceia" a base de dados dos fornecedores sem precisar interpretá-la.

Ao pensar nesta solução, foi encontrado outro problema: se for liberado para que cada fornecedor de recursos configure e libere os dados desejados, em pouco tempo a pesquisa destes recursos tornar-se-á uma tarefa árdua para o agente de viagens, pois esse teria que construir um filtro específico para cada fornecedor e depois comparar os resultados recebidos das diversas pesquisas realizadas. Pensando neste novo problema foi decidido padronizar os campos possíveis de serem liberados. Assim todos os fornecedores poderão configurar a localização destes dados em seu sistema, e caso a necessidade de liberação de um determinado dado da base do fornecedor for muito grande, é necessário apenas incluir este campo na lista de campos possíveis de serem parametrizados, liberando o mesmo para todos os outros fornecedores.

Para configurar a localização dos dados dos campos padronizados, é necessário que o fornecedor eleja uma das tabelas de sua base como principal. Esta tabela principal deve ser a tabela que possui os dados principais dos cômodos a serem liberados para a pesquisa. Feito a escolha da tabela principal poderá ser definido a localização dos outros campos contidos na lista padrão, e caso este não pertença à tabela padrão, será necessário informar a ligação da tabela a qual este campo pertence com a tabela principal do sistema.

4 CONCLUSÕES E SUGESTÕES

Com o desenvolvimento do trabalho foi possível atingir todos os objetivos propostos. Para que estes objetivos pudessem ser alcançados foi necessário, no decorrer do trabalho, um estudo mais aprofundado da tecnologia empregada, visto que se trata de uma tecnologia nova e ainda pouco utilizada no meio acadêmico mas com grandes expectativas de uso no desenvolvimento de sistemas corporativos.

Os *Web Services* demonstraram algumas qualidades muito interessantes, dentre elas:

a) reutilização de código: um mesmo *Web Service* pode ser utilizado para compor outros *Web Services;*
b) independência de plataforma e linguagem de programação: a aplicação cliente que utilizará os seus métodos pode ser escrita sem precisar se preocupar com compatibilidade nestes itens;
c) independência de banco de dados: a aplicação cliente não precisa saber qual o banco de dados que o fornecedor utiliza, pois a resposta que vem do *Web Service* sempre será do mesmo tipo.

O Delphi mostrou-se eficaz como ferramenta de desenvolvimento para *Web Services*, e tornou-se muito útil principalmente para desenvolvedores habituados a utilizar esta linguagem para o desenvolvimento de sistemas.

Como contribuição para os usuários o sistema permite que os fornecedores de recursos, os disponibilizem para um maior número de agentes de viagens. Já para os agentes de viagens a pesquisa por recursos disponíveis tornou–se mais eficaz, além de permitir que seja feita uma comparação de preços antes de efetuar qualquer reserva. Indiretamente a pessoa que está procurando reservar algum recurso também foi favorecida pois com este sistema será possível ter um maior número de opções para escolha e uma maior garantia que o recurso pesquisado esteja disponível na data solicitada.

4.1 SUGESTÕES

Como este trabalho aborda um tema relativamente novo e ainda em amadurecimento, alguns itens mencionados nesta pesquisa podem ser objetos de trabalhos futuros. Podem ser citados, entre eles:

a) desenvolvimento de aplicações *.Net* com a utilização de *Web Services;*
b) aplicação de regras de segurança nas mensagens enviadas pelas aplicações *Web Services;*
c) portal de serviços na WEB.

5 REFERÊNCIAS BIBLIOGRÁFICAS

BOX, Don; EHNEUBUSKE, David; KAKIVAYA, Gopal. **SOAP:**Simple Object Access Protocol. W3C, 2000. Disponível em: <http://www.w3.org/TR/SOAP/>. Acesso em 15 abr. 2003.

BRAY, T.; PAOLI, J.; SPERBERG-MCQUEEN, C. M. e MALER, E. **Extensible Markup Language (XML) 1.0.** Disponível em: <http://www.w3.org/TR/2000/REC-xml-20001006>. Acesso em 10 abr. 2003.

CANTÚ, Marcos. **Internet programming with Delphi.** London: Sybex, 2001a.

CANTÚ, Marcos. **Mastering:** Delphi 6. London: Sybex, 2001b.

CLEMENTS, T. **Overview of SOAP web services** – Technical Overviews**.** Sun Microsystems, August 2001.

GRAHAM, Steve; SIMEONOV, Simeonov; BOUBEZ, Toufic. **Building web services with Java.** New York: SAMS, 2002.

HENDRICKS, Mack; et al. **Professional Java web services.** London: Wrox, 2002. 588 p.

KREGER, H. **Web services conceptual architecture**. New York: IBM, 2001.

MONTANER Montejano, Jordi. **Estrutura do mercado turístico.** São Paulo: Roca, 2001.

PEREIRA, Dani Edson. **Visual Basic.Net para programadores:** evoluindo para a nova geração do VB. São Paulo: Books, 2002.

SAGANICH, A. **Java and w*eb services*.** Disponível em: <http://www.onjava.com/pub/a/onjava/2001/08/07/webservices.html>. Acesso em 15 abr. 2003.

SNELL, James; TIDWELL, Doug; KULCHENKO, Pavel. **Programming Web Services with SOAP:** Building Distributed Applications**.** O'Relly, 2001. 216 p.

SEELY, S. **SOAP cross platform web service development using XML.** New York: Prentice Hall PTR, 2000. 400 p.

TOMELIN, Carlos Alberto. **Mercado de agências de viagens e turismo:** como competir diante das novas tecnologias. São Paulo: Aleph, 2001.

WATSON, Karli et al. **Beginning C# -** programando. São Paulo: Makron, 2002.

APÊNDICE

APÊNDICE 1 - DICIONÁRIO DE DADOS DA APLICAÇÃO FORNECEDORA

Configuração das ligações dos campos

Lista de colunas

Nome	Código	Tipo	P	M
Código da ligação	COD_LIGACAO	INTEGER	Sim	Sim
Código do campo	COD_CAMPO	INTEGER	Não	Sim
Descrição do campo	DES_CAMPO	VARCHAR(80)	Não	Não
Descrição do campo de ligação	DES_CAMPOLIG	VARCHAR(80)	Não	Não

Configuração dos banco de dados

Lista de colunas

Nome	Código	Tipo	P	M
Código da configuração	COD_CONFIG	INTEGER	Sim	Sim
Indica o banco configurado	COD_DATABASE	INTEGER	Não	Não
Código do Dialeto	COD_SQLDIALECT	INTEGER	Não	Não
Tabela principal da base	DES_TABELA	VARCHAR(80)	Não	Não
Campo da tabela principal	DES_CAMPO	VARCHAR(80)	Não	Não
Nome da base de dados	DES_DATABASENOME	VARCHAR(80)	Não	Não
Charset da base	DES_CHARSET	VARCHAR(10)	Não	Não
Usuário da base	DES_USERNOME	VARCHAR(30)	Não	Não
Senha do usuário	DES_PASSWORD	VARCHAR(30)	Não	Não
Nome do servidor	DES_SERVERNOME	VARCHAR(30)	Não	Não
SQL gerado	DES_SQL	VARCHAR(500)	Não	Não
Flag autenticação via SO	FLG_AUTENTICACAOOS	VARCHAR(01)	Não	Não

Configuração dos campos liberados

Lista de colunas

Nome	Código	Tipo	P	M
Código do campo	COD_CAMPO	INTEGER	Sim	Sim
Descrição dos requisitos	DES_DESCRICAO	VARCHAR(80)	Não	Não
Descrição da tabela	DES_TABELA	VARCHAR(80)	Não	Não
Descrição do campo	DES_CAMPO	VARCHAR(80)	Não	Não
Seleção especial	DES_SELECAOESPECIAL	VARCHAR(500)	Não	Não
Flag possui imagem	FLG_IMAGEM	VARCHAR(01)	Não	Não

Requisições

Lista de colunas

Nome	Código	Tipo	P	M
Código da requisição	COD_REQUISICAO	INTEGER	Sim	Sim
Data da inclusão	DAT_REQUISICAO	DATE	Não	Não
Identificador do agente	NRO_AGENTE	VARCHAR(10)	Não	Não
Data entrada	DAT_ENTRADA	DATE	Não	Não
Data saída	DAT_SAIDA	DATE	Não	Não

Itens da requisição

Lista de colunas

Nome	Código	Tipo	P	M
Código da requisição	COD_REQUISICAO	INTEGER	Sim	Sim
Código do item	COD_ITEM	INTEGER	Sim	Sim
Descrição do recurso	DES_RECURSO	VARCHAR(50)	Não	Não
Flag autorizado	FLG_AUTORIZADO	VARCHAR(01)	Não	Não
Data autorização	DAT_AUTORIZACAO	DATE	Não	Não
Motivo	DES_MOTIVO	VARCHAR(500)	Não	Não

APÊNDICE 2 – DICIONÁRIO DE DADOS DA APLICAÇÃO

CLIENTE

Pessoa

Lista de colunas

Nome	Nome campo	Tipo	P	M
Código da pessoa	COD_PESSOA	INTEGER	Sim	Sim
Noome da pessoa	DES_PESSOA	VARCHAR(80)	Não	Não
Endereço	DES_ENDERECO	VARCHAR(80)	Não	Não
Cep	DES_CEP	VARCHAR(10)	Não	Não
Cidade	DES_CIDADE	VARCHAR(80)	Não	Não
Estado	DES_ESTADO	VARCHAR(2)	Não	Não
CPF ou CGC	DES_CPFCGC	VARCHAR(20)	Não	Não
Data cadastro	DAT_CADASTRO	DATE	Não	Não
Telefone	DES_TELEFONE	VARCHAR(20)	Não	Não
Flag tipo de pessoa	FLG_PESSOA	VARCHAR(1)	Não	Não

Clientes

Lista de colunas

Nome	Nome campo	Tipo	P	M
Código da pessoa	COD_PESSOA	INTEGER	Sim	Sim
Data nascimento	DAT_NASCIMENTO	DATE	Não	Não
Última requisição feita	DAT_ULTIMA_REQUISICAO	DATE	Não	Não

Fornecedores

Lista de colunas

Nome	Nome campo	Tipo	P	M
Código da pessoa	COD_PESSOA	INTEGER	Sim	Sim
Observacao sobre o fornecedor	DES_FORNECEDOR	VARCHAR(500)	Não	Não
URL Web service	DES_URL	VARCHAR(200)	Não	Não

Filtros

Lista de colunas

Nome	Nome campo	Tipo	P	M
Código do filtro	COD_FILTRO	INTEGER	Sim	Sim
Descrição do filtro	DES_FILTRO	VARCHAR(80)	Não	Não
Data criação	DAT_CRIACAO	DATE	Não	Não

Itens dos Filtros

Lista de colunas

Nome	Nome campo	Tipo	P	M
Código do filtro	COD_FILTRO	INTEGER	Sim	Sim
Código do item	COD_ITEM	INTEGER	Sim	Sim
Valor para o campo	DES_VALOR	VARCHAR(80)	Não	Não
Campo dos campos padrões	DES_CAMPO	VARCHAR(80)	Não	Não
Flag tipo de filtro	FLG_FILTRO	VARCHAR(1)	Não	Não

Requisições

Lista de colunas

Nome	Nome campo	Tipo	P	M
Código da requisição	COD_REQUISICAO	INTEGER	Sim	Sim
Data requisição	DAT_REQUISICAO	DATE	Não	Não
Código do fornecedor	COD_FORNECEDOR	INTEGER	Não	Sim
Código do cliente	COD_CLIENTE	INTEGER	Não	Sim

Itens da requisição

Lista de colunas

Nome	Nome campo	Tipo	P	M
Código da requisição	COD_REQUISICAO	INTEGER	Sim	Sim
Código dos itens	COD_ITEM	INTEGER	Sim	Sim
Recurso	DES_RECURSO	VARCHAR(500)	Não	Não
Valor do recurso	VLR_RECURSO	FLOAT	Não	Não

APÊNDICE 3 – DOCUMENTO WSDL IMPORTADO PELO DELPHI

```
unit IWSServer1;
interface
uses InvokeRegistry, SOAPHTTPClient, Types, XSBuiltIns;
type

  IWSServer = interface(IInvokable)
  ['{9FA4F4A4-AB6B-9725-3DE2-86A79F8BB431}']
    function  conectarBancoFornecedor: WideString; stdcall;
  end;
function GetIWSServer(UseWSDL: Boolean=System.False; Addr: string='';
                      HTTPRIO: THTTPRIO = nil): IWSServer;

implementation

function GetIWSServer(UseWSDL: Boolean; Addr: string; HTTPRIO:
THTTPRIO): IWSServer;
const
  defWSDL = 'http://anderson/WS/WSServer.exe/wsdl/IWSServer';
  defURL  = 'http://anderson/WS/WSServer.exe/soap/IWSServer';
  defSvc  = 'IWSServerservice';
  defPrt  = 'IWSServerPort';
var
  RIO: THTTPRIO;
begin
  Result := nil;
  if (Addr = '') then
  begin
    if UseWSDL then
      Addr := defWSDL
    else
      Addr := defURL;
  end;
  if HTTPRIO = nil then
    RIO := THTTPRIO.Create(nil)
  else
    RIO := HTTPRIO;
  try
    Result := (RIO as IWSServer);
    if UseWSDL then
    begin
      RIO.WSDLLocation := Addr;
      RIO.Service := defSvc;
      RIO.Port := defPrt;
    end else
      RIO.URL := Addr;
  finally
    if (Result = nil) and (HTTPRIO = nil) then
      RIO.Free;
  end;
end;
```

```
initialization
  InvRegistry.RegisterInterface(TypeInfo(IWSServer), 'urn:WSServerIntf-
IWSServer', 'utf-8');
  InvRegistry.RegisterDefaultSOAPAction(TypeInfo(IWSServer),
'urn:WSServerIntf-IWSServer#conectarBancoFornecedor');

end.
```