UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

UTILIZAÇÃO DA TECNOLOGIA *.NET* NA CONSTRUÇÃO DE SISTEMAS B2B

RODRIGO LINHARES

BLUMENAU
2004

2004/2-10

RODRIGO LINHARES

UTILIZAÇÃO DA TECNOLOGIA *.NET* NA CONSTRUÇÃO DE SISTEMAS B2B

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Sistemas de Informação — Bacharelado.

Prof. Alexander Roberto Valdameri

BLUMENAU
2004

2004/2-10

UTILIZAÇÃO DA TECNOLOGIA *.NET* NA CONSTRUÇÃO DE SISTEMAS B2B

Por

RODRIGO LINHARES

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Alexander Roberto Valdameri – Orientador, FURB
Membro:	Prof. Everaldo Arthur Grahl, FURB
Membro:	Prof. Ricardo Alencar de Azambuja, FURB

Blumenau, 30 de novembro de 2004

Quando as coisas não acontecem do jeito
que nós queremos, é porque as coisas
vão acontecer melhor do que pensamos.

AGRADECIMENTOS

À Deus.

À minha família, que sempre me apoiou e me incentivou.

À minha namorada, por sua compreensão, carinho e por me suportar nos momentos mais difíceis.

Aos amigos por idéias, sugestões e companheirismo.

Ao professor Alexander Roberto Valdameri, que acreditou na conclusão deste trabalho e sempre mostrou-se interessado pela arte de ensinar.

E a banda AC/DC que me motiva em todos os momentos.

RESUMO

O presente trabalho apresenta a implementação de um aplicativo que possibilita a comunicação entre aplicações de comércio eletrônico, utilizando a *internet* como meio. A aplicação utiliza o conceito de *Web Service* como tecnologia baseada no protocolo SOAP e na linguagem XML, em um servidor HTTP. As aplicações foram implementadas para utilizar o recurso, afim de demonstração das tecnologias.

Palavras chave: *Web Service*, SOAP, XML.

ABSTRACT

The present work presents the implementation of an application that makes possible the communication between e-commerce applications, using the internet as a way. The application utilizes the concept of *Web Service* as a technology based on a SOAP protocol and XML language, in a HTTP server using. The applications were implanted to use the recourse in order to the demonstration of technologies.

Key-words: *Web Service*, SOAP, XML.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

B2B - *Business to Business*

B2C - *Business to Consumer*

CLR - *Common Language Runtime*

C# - *C-Sharp*

DLL - *Dynamic Link Library*

HTTP - *Hypertext Transfer Protocolo*

HTML - *Hypertext Markup Language*

IL - *Intermediate Language*

RAD - *Rapid Application Developmen*

SGML - *Standart Generalized Markup Language*

SOAP - *Simple Object Access Protocol*

SQL - Structured Query Language

UML - *Unified Modeling Language*

XML - *Extensible Markup Language*

WSDL - *Web Service Description Language*

W3C - *World Wide Web Consortium*

Sumário

1 INTRODUÇÃO

1.1 CONTEXTUALIZAÇÃO

No início dos anos 90 a *internet* começou a ser utilizada de forma comercial, tornando-se conhecida como uma tecnologia de grande importância e potencialidade para mudanças nas organizações. Franco (2001) explica que o seu uso para fins comerciais de forma maciça começou com o modelo de venda direta aos consumidores – B2C (*business-to-customer*). No entanto, um novo modelo de relacionamento externo surgiu e está se mostrando em dimensões incalculáveis: o negócio entre empresas pela *internet* – B2B (*business-to-business*). Estes dois modelos formam o que é conhecido como *e-Commerce* ou comércio eletrônico.

No desenvolvimento de sistemas voltados para o B2B, para haver a integração entre os vários sistemas envolvidos, muitos fatores devem ser levados em conta. Um dos fatores que merece atenção é a forma de comunicação entre os sistemas.

A tecnologia *Web Services* promete ser o alicerce para a troca de informações entre aplicativos em um ambiente de computação distribuída. Os *Web Services* trabalham como interfaces de funcionalidades dos aplicativos disponíveis em *Web*, sendo esta interface pública, é então conhecida por qualquer um. Os *Web Services* utilizam-se do protocolo *Simple Object Access Protocol* (SOAP) para descrever o formato para a troca de dados com outros componentes. Os pacotes são baseados no padrão *eXtensible Markup Language* (XML). XML e SOAP são dois protocolos que atualmente estão sendo amplamente usados como padrão em diversos aplicativos para *internet*.

Para o desenvolvimento deste projeto foi utilizada a plataforma *.NET* da Microsoft. A plataforma busca simplificar a introdução ao conceito de *Web Services*. Bragnolo (2004) explica que o *.NET Framework* implementa muitas funcionalidades para facilitar o desenvolvimento de aplicativos *Web*. Para isso, o *.NET* incorpora e possui total suporte aos *Web Services*, um recurso que introduz um novo conceito de computação, já que permite aos desenvolvedores chamar métodos e escrever componentes via *internet*, independentemente da plataforma ou linguagem.

Este trabalho utiliza o conceito de *Web Services* através da tecnologia *.NET* para uma aplicação B2B. Para validar os estudos desta tecnologia, utiliza-se o ambiente que

envolve sistemas Hoteleiros. Os mesmos oferecem condições reais de aplicabilidade, uma vez que vislumbram a possibilidade de disponibilizar informações e serviços de interesse para agências de turismo, sem que haja qualquer dependência funcional entre as duas entidades. Além da aplicação para a integração das informações, serão desenvolvidos dois módulos, um módulo diz respeito ao ambiente envolvendo o hotel que ilustra um sistema hoteleiro, e o outro ambiente diz respeito ao ambiente envolvendo um sistema de agências de turismo. Existe uma integração envolvendo estes dois ambientes através do *Web Service* proposto para o funcionamento da troca de informações.

1.2 OBJETIVOS

Este trabalho tem como objetivo principal construir mecanismos para prover a troca de informações e a divulgação de serviços de hotéis, permitindo a integração com possíveis parceiros.

Os objetivos específicos do trabalho são:

a) criação de um *Web Service* que permita que os hotéis disponibilizem informações sobre seus serviços;
b) criação de um módulo que simule a interface de um hotel;
c) criação de um módulo que simule uma agência de turismo;
d) demonstrar com os dois módulos criados a interoperabilidade da troca de informações.

1.3 MOTIVAÇÃO

Este trabalho motiva-se na construção de um mecanismo que utilize protocolos padrões de comunicação para integrar dois sistemas distintos, garantindo independência de plataforma e simplicidade na comunicação.

A simplicidade da integração entre aplicativos pode tornar-se um diferencial competitivo em meio às empresas que utilizam a *internet* como meio de troca de informação.

Para suporte à tecnologia de *Web Services* foi utilizado o protocolo SOAP juntamente com XML, ambas soluções emergentes no desenvolvimento de aplicações distribuídas.

1.4 ESTRUTURA DO TRABALHO

Este trabalho apresenta a fundamentação teórica nos capítulos 2 e 3, que tratam as visões da construção da aplicação, comércio eletrônico e das tecnologias utilizadas para tanto. Esses capítulos destacam tecnologias como *Web Services*, XML, SOAP e a plataforma *.NET* utilizada para a criação da aplicação.

O capítulo 4 trata de tópicos relacionados com a especificação e implementação da aplicação, contendo os requisitos principais, diagramas, especificação e implementação da solução. Para ilustrar o funcionamento é apresentado um ambiente caracterizado para o estudo de caso.

Finalizando com a conclusão, dificuldades encontradas na elaboração do projeto e sugestões para trabalhos futuros, descritos no capítulo 5.

2 APLICAÇÕES PARA INTERNET

Neste capítulo serão apresentados os tópicos sobre Comércio Eletrônico e Tipos de aplicações baseados no ambiente *Web*.

2.1 COMÉRCIO ELETRÔNICO

O ambiente empresarial nos últimos tempos tem passado por drásticas mudanças operacionais. Muitas dessas mudanças estão ligadas diretamente com a tecnologia de informação. Este novo ambiente empresarial que vem surgindo caracteriza-se em globalização, integração de informações e facilidade de comunicação. Estas características têm confirmado a tendência da utilização de meios eletrônicos como mecanismos de troca de informação.

Abertin (2000) explica que o comércio eletrônico é a realização de toda a cadeia de valor dos processos de negócio num ambiente eletrônico, por meio da aplicação intensa das tecnologias de comunicação e de informação atendendo os objetivos de negócio. Os processos incluem transações *business-to-business*, *business-to-customer* e intra-organizacional.

Kalakota (2002) explica que o impacto do *e-Commerce* acontece em etapas. Em sua primeira etapa, as empresas iniciam suas participações na *internet*, para tornarem-se conhecidas, sem saber o motivo, porém cientes da necessidade de estar *on-line*. Numa segunda etapa entra as transações, compra e venda por meio da *internet*. Nesta etapa as empresas começaram a descobrir que estavam ganhando cada vez mais concorrentes, ou melhor, conhecendo-os por este novo canal de acesso. Atualmente esta última etapa está voltada na *internet* como influenciador da lucratividade. Esta fase chamada de *e-Business*, inclui todas aplicações e os processos que permitem a uma empresa realizar transações.

O *e-Business* trata-se de uma estratégia global de redefinição dos antigos modelos de negócios, com o auxílio da tecnologia que envolve transações de comércio eletrônico ou de compras e vendas pela internet, buscando valorizar o cliente e maximizar os lucros.

O comércio eletrônico está permitindo que companhias tornem-se mais eficientes e flexíveis em suas operações, trabalhando mais próximos de clientes e fornecedores, sendo mais ágeis às necessidades e expectativas. A localização geográfica

deixa de ser uma barreira para o comércio; o comércio eletrônico torna-se, todavia, um facilitador relevante na economia mundial.

De acordo com Abertin (2000), o comércio eletrônico não pode atingir todo seu potencial como um conceito isolado, considerando apenas cliente-organização, organização-organização, ou atividade de automação interna, desconectadas. Para as empresas serem completamente efetivas, essas três atividades precisam estar integradas. Atualmente, a ênfase da vantagem competitiva de uma empresa está nas capacidades que permitem a um negócio entregar consistentemente um valor superior para seus clientes ou parceiros por meio de melhor coordenação e gerenciamento de fluxo de trabalho e gerenciamento de produtos e serviços.

InfoBussines (2004) ressalta que o impacto do comércio eletrônico será profundo nas empresas e na sociedade como um todo. Para aquelas companhias que exploram plenamente o seu potencial, o comércio eletrônico oferece a possibilidade de mudanças de paradigmas, mudanças que tão radicalmente alteram as expectativas dos clientes e redefinem o mercado ou criam mercados totalmente novos.

2.2 TIPOS DE APLICAÇÃO

Aplicações existentes para o comércio eletrônico são várias. Uma utilidade comum do comércio eletrônico é a troca de informação de negócio sem o uso do papel, utilizando os diversos recursos eletrônicos para tal.

No escopo de atuação no comércio eletrônico, existem três categorias possíveis: Negócio-a-Negócio (B2B – *Business to Business*), Negócio-a-Consumidor (B2C – *Business to Consumer*) e o ambiente intra-organizacional.

A integração de processos de diferentes empresas necessita de padronização de protocolos de comunicação. Franco (2001) explica que a necessidade de agilização e facilidade de integrar um novo parceiro no processo é uma variável determinante. Visto dessa maneira, é necessário o uso de um protocolo de comunicação universal, padronizado e de grande aceitação por todos os participantes das cadeias de serviços. O uso da internet com ferramentas chaves passam a alavancar enorme vantagem competitiva, pois podem responder a este requisito.

2.2.1 Aplicação B2C

A categoria negócio-a-consumidor é a mais comum, envolvendo uma organização e os clientes finais, baseado puramente na *internet*. Albertin (2000) explica que nas transações negócio-a-consumidor eletronicamente facilitadas, os clientes aprendem sobre produtos por meio de publicação eletrônica, compram produtos com dinheiro eletrônico e outros sistemas de pagamentos seguros.

A compra de produtos e serviços pela internet está causando uma revolução na vida dos consumidores, como também no mundo dos negócios. Torna-se muito mais cômodo fazer compras, reservar hotéis ou reservar passagens aéreas pela *internet*, do que se deslocar fisicamente.

Tem-se como exemplo a empresa aérea TAM (endereço eletrônico http://www.tam.com.br), onde qualquer pessoa pode fazer consultas sobre passagens e comprar o bilhete aéreo. Isto trata-se de um claro exemplo de uma aplicação *e-Commerce* do tipo B2C. O exemplo é ilustrado na figura 1, que mostra a compra de um bilhete aéreo de Navegantes para Foz do Iguaçu, diretamente da página da companhia.

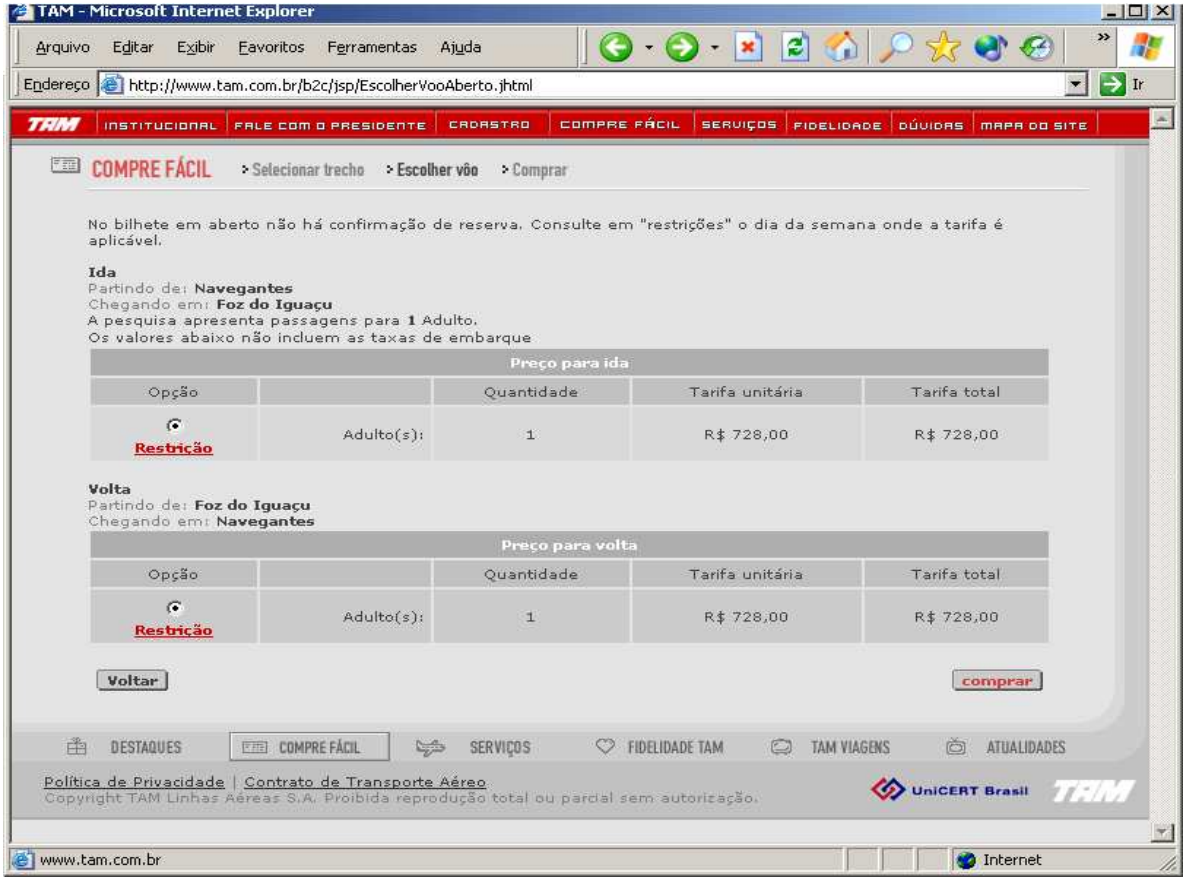


Figura 1: Compra on-line de bilhetes aéreos (www.tam.com.br).

2.2.2 Aplicação B2B

A categoria negócio-a-negócio envolve duas organizações, representando um processo de compra e venda entre uma organização e seus fornecedores ou parceiros. Franco (2001) explica que o modelo B2B, embora também esteja baseado na i*nternet*, usa predominantemente os recursos de *extranet*. Empresas distintas abrem acesso de suas redes às redes de parceiros, quer sejam fornecedores, prestadores de serviços ou canais de distribuição, formando redes de acessos remotos com troca de informações em ambos os sentidos.

Em muitas das transações efetuadas pelo *e-commerce* (compras via *internet*), há em seu interior o que é conhecido como *e-business*, ligação entre um sistema de cobrança bancária, cartão de crédito, entre outros. O *e-business* torna-se nada mais que um sistema que faz a interligação de diversas empresas, interagindo, para que o *e-commerce* aconteça. Pode-se dizer que o *e-commerce* é a parte visível do *e-business.*

Em geral as empresas que pretendem desenvolver uma estratégia competitiva em um determinado setor, identificam atividades que são importantes para todas e que pode ser executada conjuntamente, sem prejuízos de competitividade. Um exemplo é o *Covisint*, portal global de compras da *General Motors*, *Ford* e *Daimler-Chrysler*, que já recebeu a adesão da *Renault* e da *Nissan*. O objetivo é reduzir custos, controlando melhor os estoques e agilizar o processo de compras. (Franco, 2001)

Outro exemplo pode ser encontrado nas buscas de ofertas de vôos na *internet*, onde um portal executa uma pesquisa para algum cliente interessado listando todas as companhias aéreas que disponibilizem o serviço de acordo com aquilo que o cliente deseja. Estes portais trabalham em conjunto com diversas companhias aéreas simultaneamente, encontrando desta maneira uma nova estratégia competitiva. No exemplo demonstrado na Figura 2, foi feita uma consulta de vôos de Atlanta com destino a Miami, no dia 9 de novembro às 20:00 horas, e retornando à cidade de origem no dia 12 de novembro no mesmo horário.

Como resultado desta pesquisa, tem-se uma lista com as diversas companhias aéreas que estão disponibilizando o serviço de acordo com o informado no exemplo.

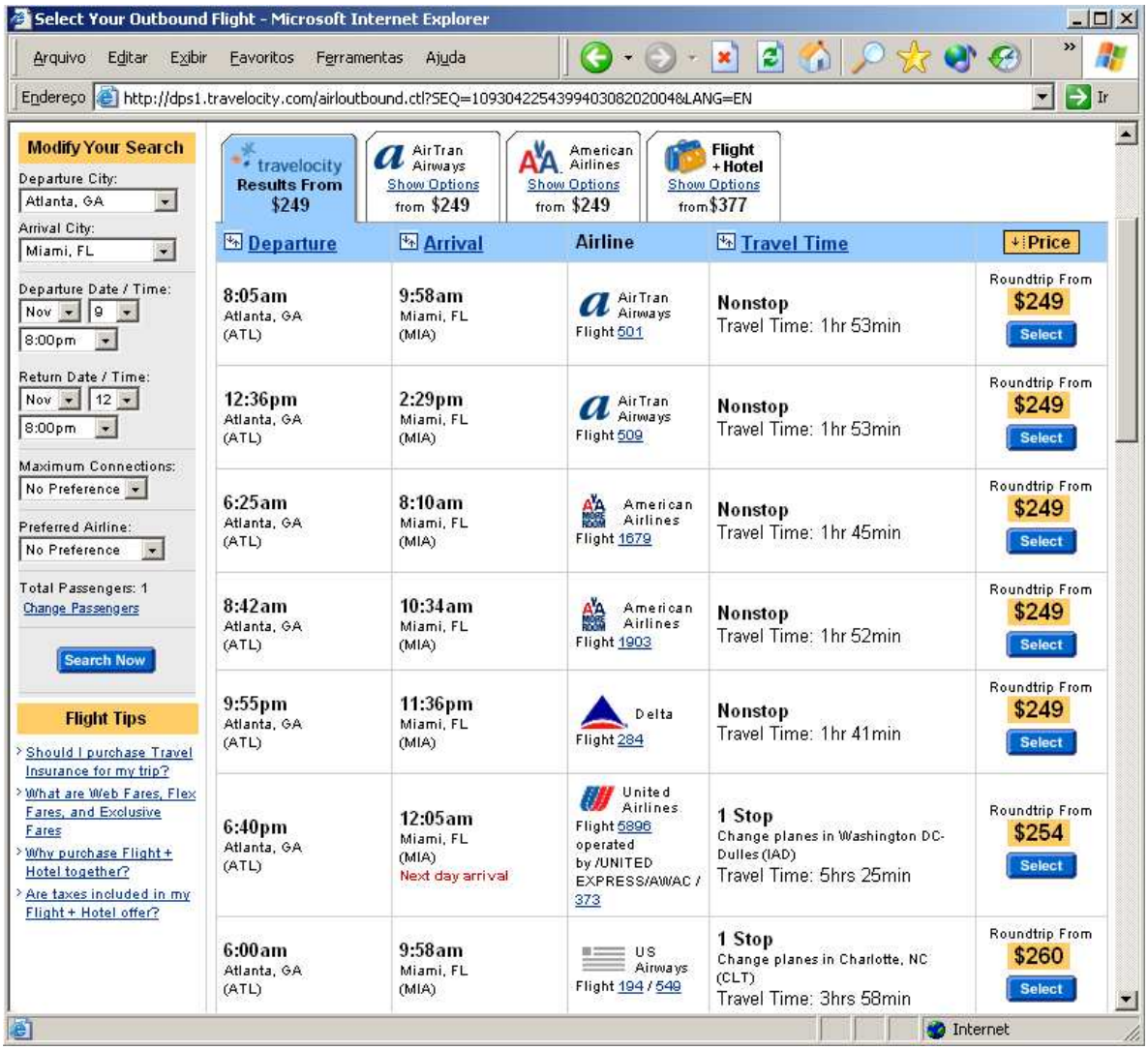


Figura 2: Ofertas de bilhetes aéreos por companhia (www.travelocity.com).

3 COMPONENTES DISTRIBUÍDOS

Para se conseguir gerenciar a complexidade e as mudanças inerentes de sistemas voltados para o *e-commerce*, um pacote de componentes distribuídos torna-se necessário. Para se construir e operar cenários avançados de integração cliente/servidor, *internet* e *Intranets*, pressupõe-se que a solução de objetos distribuídos é a melhor escolha.

De acordo com Microsoft (2002), objetos distribuídos ou *business objects*, reduzem muita a complexidade de desenvolvimento das soluções, pois os desenvolvedores não necessitam conhecer o interior do objeto e de que forma este objeto trabalha; apenas precisam conhecer a sua interface, para saber quais são os serviços disponibilizados. Os objetos podem ser distribuídos dinamicamente para uso e reuso, além de manter o máximo de funcionalidades de cada sistema sem que haja redundâncias. É desenvolvido para múltiplas aplicações e para plataformas heterogêneas e fazem parte de uma arquitetura de *software* em camadas.

3.1 *WEB SERVICES*

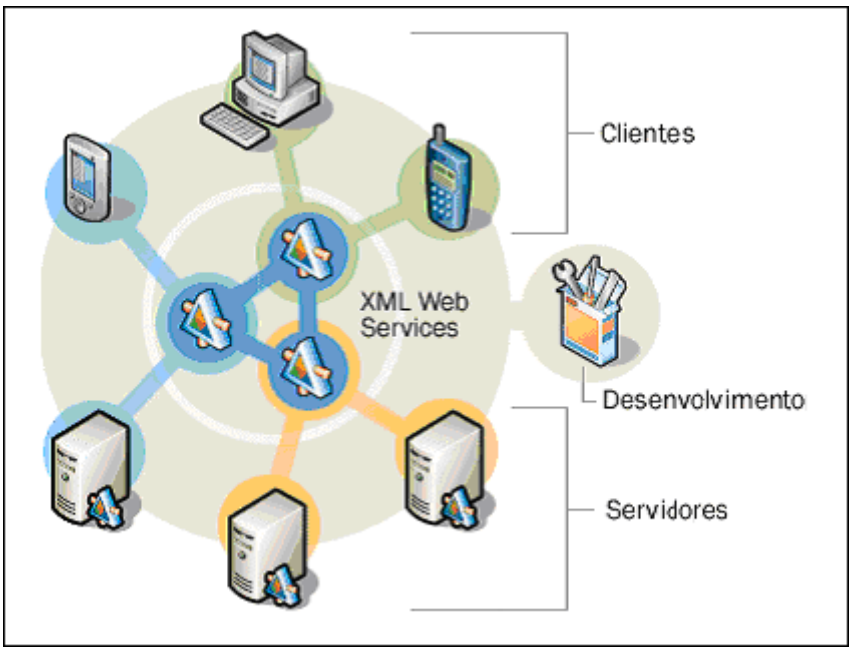
Os *Web Services* XML oferecem os caminhos para que as empresas possam trocar informações entre aplicativos de uma forma simples, segura e robusta. Os *Web Services* possibilitam que softwares em qualquer tipo de computadores ou dispositivos móveis se conectem e interoperem entre si, seja dentro de uma mesma empresa ou com parceiros externos e clientes. Através de um meio de conexão padronizado, a integração torna-se mais rápida, fácil e flexível, podendo ser executada entre diferentes sistemas operacionais, linguagens de programação e tecnologias (Bragagnolo, 2003).

A idéia principal dos *Web Services* propõe que os desenvolvedores criem aplicações a partir de serviços e objetos disponíveis na *Web*. Para Tolomelli (2004), *Web Services* são componentes de *software* que são chamados a partir de outros aplicativos na *internet*. É a tecnologia que permite que computadores na *internet* conversem entre si sem a intervenção direta dos usuários. Portanto, *Web Services* são serviços e objetos (aplicação lógica) disponibilizados em um servidor *Web*, permitindo que qualquer aplicação possa acessar suas funções independente da similaridade entre

os sistemas envolvidos, Sistemas Operacionais ou implementações internas de linguagem. Com o uso de protocolos padrões, a comunicação entre aplicações torna-se transparente.

Snell, Tidwell e Kulchenko (2002) explicam que *Web Services* são interfaces posicionadas entre o código da aplicação e o código do usuário. Ele atua em uma camada de abstração, separando a plataforma e detalhes de desenvolvimento do código da aplicação que invoca o serviço. O nível de abstração que oferece os recursos envolvendo *Web Services* torna independente a linguagem na qual serão escritos os programas clientes, isto é, os que farão a comunicação com o servidor. Pode-se dizer que a utilização de *Web Services* permite a interoperabilidade entre plataformas.

A figura 3 ilustra a idéia de integração de diversos dispositivos com a *internet* através de *Web Services*.



Fonte: Microsoft (2002).
Figura 3: Esquema ilustrativo da tecnologia *Web Services.*

Os *Web Services* baseiam-se no protocolo SOAP (*Simple Object Access Protocol*) para comunicação e utilizam arquivos XML para tráfego das informações. Qualquer plataforma que interprete rotinas HTTP e manipule XML pode utilizar os dados dos *Web Services*.

3.1.1 Estrutura dos *Web Services*

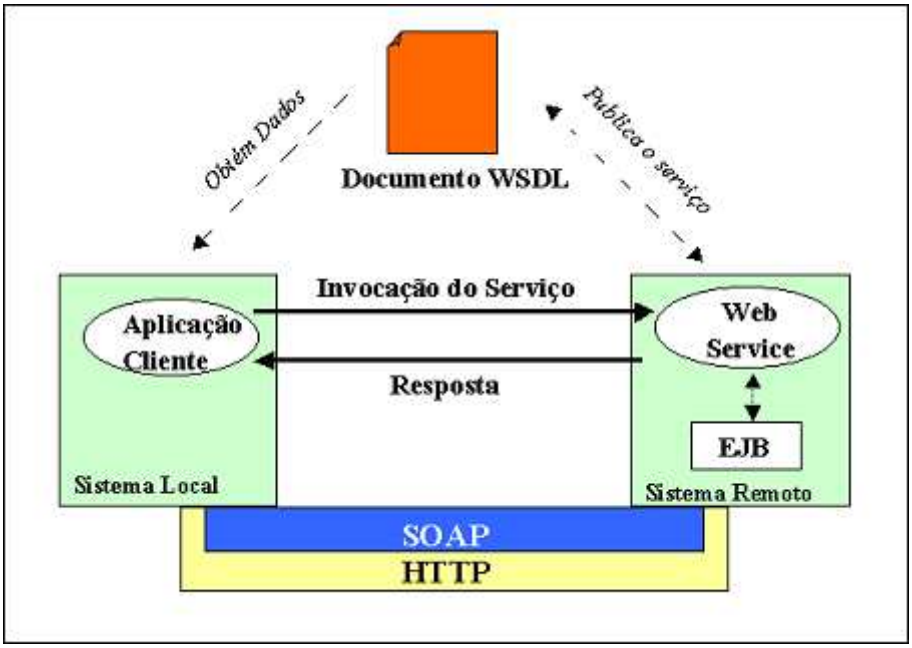
Os Web Services dispõem de uma infra-estrutura muito bem organizada no seu funcionamento. Bragagnolo (2004) apresenta a infra-estrutura dos *Web Services* da seguinte forma:

- *Web Services Directories:* é um diretório central para fazer a localização de *Web Services* oferecidos por outras organizações. Um exemplo de diretório desse tipo é o www.uddi.org. Não necessariamente, um cliente *Web Service* precisa referir-se a um *Web Services Directory*.

- *Web Services Discovery:* trata-se de um mecanismo de descoberta para localizar *Web Services*. Esse mecanismo utiliza o WSDL. O *Web Service* possui uma especificação chamada DISCO (de "*DISCOvery"*) que define um algoritmo para localizar descrições de serviços. Se uma aplicação já conhece a localização da descrição do *Web Service*, este processo de descoberta torna-se desnecessário.

- *Web Services Description:* para haver comunicação entre uma aplicação e um *Web Service*, e que saibam interagir um com o outro, o *Web Service* precisa prover uma descrição de seus serviços, que define as interações suportadas. Os clientes de *Web Services* devem saber interagir com um *Web Service* antes de usá-lo.

- *Web Services Wire Formats:* para permitir comunicação universal, os *Web Services* utilizam formatos abertos de comunicação, que consistem em protocolos reconhecidos por qualquer sistema capaz de suportar os padrões mais comuns da *internet*. O protocolo SOAP *(Simple Object Access Protocol)* é a chave para a comunicação dos *Web Services.*



Fonte: Bragagnolo (2004).
Figura 4: Funcionamento de um *Web Service*.

3.1.2 WSDL

Quando um *Web Service* é implementado, faz-se necessário que seus serviços sejam conhecidos, para que os aplicativos clientes possam chamar e usar. O *Web Service Description Language* (WSDL) é um padrão que descreve os serviços implementados no *Web Service*.

Para Snell, Tidwell e Kulchenko (2002), com o fornecimento destas informações detalhadas, torna-se fácil para um usuário construir uma aplicação cliente que utilize tais funcionalidades. Esta descrição é fornecida no formato XML, gerado automaticamente pelo *.NET* , e também é utilizado para validar as chamadas ao próprio *Web Service*.

De acordo com Bragagnolo (2004), o arquivo WSDL assemelha-se a um contrato legal, já que descreve exatamente como as duas partes (provedor de *Web Services* e consumidor de *Web Services*) deverão comunicar-se e cooperar entre si. Além disso, especifica o formato de mensagem esperado pelo *Web Service*. O WSDL preserva o consumidor do trabalho interno ocorrido nos *Web Services*.

3.2 XML

Em meados da década de 90, um grupo de empresas e organizações denominado *World Wide Web Consortium* (W3C) começou a trabalhar em uma linguagem de marcação que combinava a flexibilidade da *Standard Generalized Markup Language* (SGML), com a simplicidade da HTML. O XML possui um padrão aberto, que é controlado pela W3C como uma recomendação formal; um documento que descreve o que é e como deve ser usado (RAY, 2001).

De acordo com Holzner (2001), o SGML é uma linguagem de marcação com enormes capacidades, mas também muito complexa. O XML surge como um subconjunto da SGML, porém com mais facilidade de manuseio.

O XML é um formato universal que é usado para descrever e formatar documentos estruturados e informações na *internet*. O XML define a estrutura de dados de uma maneira aberta e autodescritiva. Esta maneira de autodescrição é de forma aberta, e permite que dados sejam facilmente transferidos sobre a rede e consistidos pelo receptor. O documento XML descreve como os dados são estruturados, e não como eles devem ser mostrados ou usados, similar ao *Hypertext Markup Language* (HTML). Documentos XML contêm *tags* que designam significado para o conteúdo, permitindo aos programadores acharem um determinado dado no mesmo (Microsoft, 2002).

De acordo com Microsoft (2002), as principais partes do XML incluem instruções de processamento, elementos, atributos e comentários.

A seguir são exploradas cada uma destas partes.

3.2.1 Instrução de Processamento

A instrução de processamento é uma forma de fornecer informações a uma aplicação. Estas instruções não são textualmente parte do documento XML, mas são necessários para consolidar o documento para o processador do XML, sendo esta informação opcional para o documento (Sousa, 2004).

3.2.2 Elementos

A forma mais comum de marcação é através de elementos. Estes são delimitados pelos sinais de menor e maior, a maioria dos elementos identificam a natureza do conteúdo que envolve. Se um elemento não é vazio, ele inicia com uma marca de início , **<Elemento>**, e termina com uma marca de término, **</Elemento>.** Cada documento XML tem somente um elemento de nível mais alto, conhecido como elemento documento. Possuem um nome que os identifica e podem conter descendentes. Estes descendentes podem ser outros elementos, instruções de processamento ou comentários (Sousa ,2004)**.**

```
- <Pessoas>
  - <Pessoa>
      <Nome>Rodrigo Linhares</Nome>
      <Idade>21</Idade>
    </Pessoa>
  </Pessoas>
```

Quadro 1: Elementos XML contendo dados.

3.2.3 Atributos

Segundo Skonnard (2002), qualquer elemento poderá conter atributos. Atributos nada mais é que uma forma de dotar informações sobre os elementos. Na verdade, são pares de valores nomeados que ocorrem dentro das marcas de início, após o nome do elemento. Pode-se dizer que os atributos estão descritos como uma forma de prover informações sobre um elemento. Um elemento pode conter qualquer número de atributos, desde que todos tenham nomes diferentes.

```
- <Pessoas>
    <Pessoa Nome="Rodrigo Linhares" Idade="22" />
  </Pessoas>
```

Quadro 2: Atributos XML contendo dados.

3.2.4 Comentários

De acordo com Sousa (2004), comentários iniciam com “**<!—**” e terminam com “**-->”,** podendo conter qualquer dado, exceto a literal "--". Os comentários não fazem parte de um conteúdo textual de um documento XML. Não é preciso um processador

específico para reconhecê-los na aplicação, sendo que os comentários são opcionais para o documento.

```
- <Pessoas>
    <Pessoa Nome="Rodrigo Linhares" Idade="22" />
    <!--  Teste de comentário inserido dentro do XML   -->
    <Pessoa Nome="Ricardo Linhares" Idade="16" />
  </Pessoas>
```

Quadro 3: Comentário dentro do XML.

3.2.5 XML Válidos

Com os dados XML válidos e bem-formatados, o documento XML torna-se autodescritivo, podendo ser usado em qualquer lugar onde a troca ou transferência de informação é necessária. Desta forma, pode-se usar o XML para descrever informações sobre páginas HTML, ou descrever dados contidos em objetos ou regras de negócios, ou ainda, transações eletrônicas comerciais (Microsoft, 2002; Sousa, 2004).

Para que ocorra a validação dos documentos XML, existe um arquivo que contém o *schema*. Um *schema* contém definições dos elementos, atributos e tipos de dados que existe dentro de um documento XML, e utiliza objetos XML válidos para descrever o conteúdo de determinado documento. O *schema* define formalmente quais elementos e quais combinações possíveis são permitidas dentro de um documento XML. Os *schemas* são utilizados pelo protocolo SOAP para especificar os elementos XML (Seely, 2002).

3.3 SOAP

De acordo com Seely (2002), o *Simple Object Access Protocol* (SOAP) é um mecanismo de comunicação interaplicação. É um protocolo simples e leve para troca de informações em um ambiente distribuído e descentralizado. O SOAP permite que dois processos comuniquem-se entre si, desconsiderando o *hardware* e a plataforma que eles estão rodando.

A função do SOAP na tecnologia de *Web Services* é ser o protocolo de empacotamento padronizado para as mensagens que trafegam entre as aplicações. As especificações definem um envelope XML para as informações na transferência, e um

conjunto de regras para o aplicativo de transação (Bragagnolo, 2004). Como um protocolo de empacotamento, para SOAP não importa qual é o protocolo de transporte para enviar as mensagens. Isso faz do SOAP um protocolo extremamente flexível.

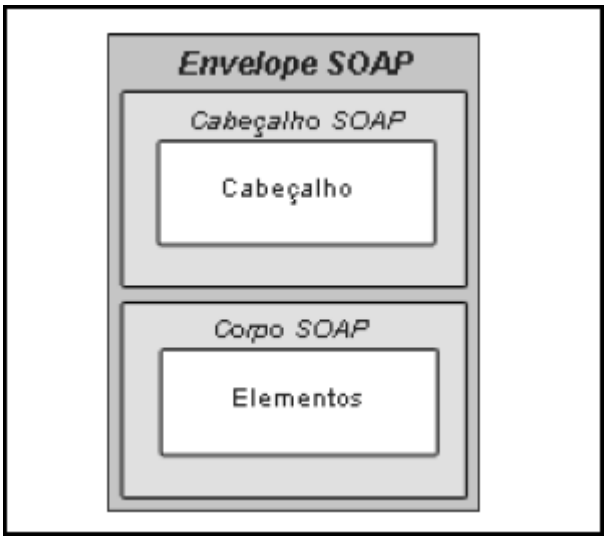
Leopoldo (2004) explica que SOAP é um protocolo que define uma gramática XML especializada, porém flexível, que padroniza o formato das estruturas das mensagens. As mensagens são, por outro lado, o método fundamental de troca de informações entre os *Web Services* e os seus consumidores. Algumas das funcionalidades do protocolo SOAP são:

- interoperabilidade entre sistemas utilizando linguagens e protocolos padronizados largamente difundidos, como XML e HTTP;
- permite a comunicação entre sistemas protegidos por *firewalls*, sem precisar abrir portas adicionais e possivelmente não seguras. Ele utiliza na maioria dos servidores, a porta 80;
- SOAP descreve completamente cada elemento na mensagem, facilitando o entendimento e a proteção contra erros.

3.3.1 Mensagens SOAP

De acordo com Snell, Tidwell e Kulchenko (2002), uma mensagem SOAP consiste em um envelope contendo um cabeçalho opcional e um corpo obrigatório. O cabeçalho contém partes de informações relevantes para o processamento da mensagem, e inclui também, rotas e formatação de entrega, autenticação ou autorização e contexto de transação. No corpo contém a mensagem que deve ser entregue e processada.

Leopoldo (2004) apresenta a estrutura do envelope SOAP da seguinte forma, evidenciado na figura 5:



Fonte: Snell, Tidwell e Kulchenko (2002)
Figura 5: Estrutura da mensagem SOAP.

3.3.2 Envelope SOAP

Na mensagem SOAP, o envelope é obrigatório e funciona para encapsular os outros elementos da mensagem, como o cabeçalho e o corpo. Assim como o nome e o endereço de uma carta entregue pelo correio, o envelope SOAP precisa das informações específicas do protocolo de transporte que está ligado a ele, com o intuito de garantir a chegada ao local certo (Scribner, 2001).

Leopoldo (2004) define que um dos principais motivos da implementação do cabeçalho no envelope SOAP, é que o objeto pode ser filtrado por *firewalls* baseados nas informações contidas nos cabeçalhos, sem consultar o XML. O quadro 4 ilustra a formatação de um envelope SOAP.

```
<soap:Envelope
  xmlns:xsi="Schema-Instance"
  xmlns:xsd="Schema"
  xmlns:soap="Envelope">
  <soap:Body>
    <!-- Os elementos do corpo inseridos aqui!!! -->
  </soap:Body>
</soap:Envelope>
```

Fonte: Leopoldo (2004).
Quadro 4: Exemplo de formatação de envelope SOAP.

3.3.3 Cabeçalho SOAP

O cabeçalho SOAP é uma parte opcional da mensagem e o conceito é similar aos *meta tags* dos documentos HTML. Eles definem meta-dados que podem prover um contexto para a mensagem ou redirecionar o processamento da mensagem (Leopoldo, 2004).

Tipicamente, as informações contidas nos cabeçalhos SOAP são utilizadas na autenticação de mensagens das requisições, ou prover mecanismos para gerenciar transações.

O quadro 5 ilustra a formatação de um cabeçalho SOAP.

```
<soap:Envelope
  xmlns:xsi="Schema-Instance"
  xmlns:xsd="Schema"
  xmlns:soap="Envelope">
  <soap:Header>
    <Autentica xmlns="Local">
      <Usuario>usuario</Usuario>
      <Senha>senha</Senha>
    </Autentica>
  </soap:Header>
  <soap:Body>
    <!-- Os elementos do corpo inseridos aqui!!! -->
  </soap:Body>
</soap:Envelope>
```

Fonte: Leopoldo (2004).
Quadro 5: Exemplo de formatação do cabeçalho SOAP.

3.3.4 Corpo SOAP

Leopoldo (2004) explica que o corpo da mensagem SOAP é obrigatório, ela carrega os dados para o protocolo SOAP e dados de uma chamada de um método particular, tais como nome do método e parâmetros de entrada, saída e resultados produzidos pelo método. As utilizações do corpo da mensagem incluem chamadas remotas a métodos e notificações de erros. Ele está presente logo após o cabeçalho da mensagem, se este existir. Caso não exista, ele deve aparecer imediatamente após a *tag* de abertura do envelope.

O conteúdo do corpo da mensagem varia de acordo com a direção da informação, podendo ser uma requisição ou uma resposta. Sendo uma requisição, dentro do corpo irá estar contida informação relevante sobre a chamada do método. Caso for uma resposta, contém dados referentes ao resultado retornado do método. Poderá estar

contido neste elemento qualquer informação que possa ser serializada, segundo o conjunto de regras definidos na especificação do protocolo.

Segundo Leopoldo (2004) a requisição é feita para converter um valor, e a resposta é o valor convertido, conforme ilustrado nos quadros 6 e 7.

```
<soap:Envelope
  xmlns:xsi="Schema-Instance"
  xmlns:xsd="Schema"
  xmlns:soap="Envelope">
  <soap:Body>
    <Converte xmlns="http://conv.fractalti.com.br">
      <Valor>10</Valor>
      <De>DEC</De>
      <Para>BIN</Para>
    </Converte>
  </soap:Body>
</soap:Envelope>
```

Fonte: Leopoldo (2004).
Quadro 6: Requisição SOAP.

```
<soap:Envelope
  xmlns:xsi="Schema-Instance"
  xmlns:xsd="Schema"
  xmlns:soap="Envelope">
  <soap:Body>
    <ConverteResponse xmlns="http://site.com.br">
      <ValorResult>1010</ValorResult>
    </ConverteResponse>
  </soap:Body>
</soap:Envelope>
```

Fonte: Leopoldo (2004).
Quadro 7: Resposta SOAP.

3.4 TECNOLOGIA *.NET*

De acordo com Microsoft (2002), a plataforma *.NET* é um modelo de desenvolvimento no qual o aplicativo torna-se um dispositivo independente da plataforma, onde dados e funcionalidades do aplicativo podem ficar disponíveis na internet. O *.NET Framework* é a infra-estrutura do *.NET*.

Criada como uma arquitetura aberta, *.NET* é uma plataforma que pode ser usada para construção e execução da nova geração de aplicativos Microsoft e aplicativos *Web*. A meta da Microsoft para a plataforma *.NET* foi simplificar o desenvolvimento para *Web*.

A plataforma *.NET* consiste destas tecnologias:

a) *.NET Framework*;

b) *.NET Enterprise Servers*;

c) *Building Block Services*;

d) *Visual Studio .NET*.

Segundo Sills *et al.* (2002), a arquitetura *.NET* é a solução para garantir uma base sólida para aplicações distribuídas. A arquitetura *.NET* faz uso intenso de arquivos XML, sendo o XML a linguagem padrão do *Framework*; não apenas para encapsular dados para tráfego entre aplicações, mas também usado para arquivos de configurações.

Bragagnolo (2004) explica que a plataforma *.NET* é um estúdio de desenvolvimento totalmente orientado a objeto. A plataforma faz com que os algoritmos de programas sejam declarados dentro de classes, e essas classes são compiladas para um formato de arquivo padrão chamado IL (*Intermediate Language*). O IL faz com que essas classes tornem-se objetos executáveis, sendo assim, podem ser herdadas diretamente a partir do executável, em qualquer linguagem da plataforma. No *Framework .NET* todos os objetos disponíveis são implementados através de classes, as quais sabem como interagir com a infra-estrutura na qual estão sendo executados.

3.4.1 *.NET Framework*

O *Framework .NET* é uma plataforma da Microsoft para desenvolvimento, visando facilidades no desenvolvimento de componentes distribuídos, interoperabilidade e multi-plataforma.

As aplicações desenvolvidas na plataforma *.NET* são independentes do Sistema Operacional, pois a arquitetura *.NET* possui sua própria biblioteca de classes, DLLs (*Dinamic Link Library*), funções e outros recursos. Essa biblioteca é chamada de *.NET Framework*. Para os desenvolvedores, o *Framework* é o próprio Sistema Operacional, disponibilizando todos os recursos necessários. É um ambiente que gerencia o desenvolvimento e execução da aplicação, onde torna-se responsável pelo

gerenciamento de todos os aspectos de execução do programa. Possuem dois principais componentes :

- CLR (*Common Language Runtime)*: responsável pelo gerenciamento da execução dos códigos.
- *.NET Framework Class Library*: coleção de classes úteis que podem ser reutilizadas, e estão integradas diretamente ao CLR.

3.4.2 Aplicação da Plataforma *.NET*

Um dos pontos fortes da plataforma *.NET* é a sua ferramenta *Rapid Application Development* (RAD) para desenvolvimento *Web*, o *ASP .NET*. Com essa ferramenta, espera-se que a produtividade no desenvolvimento para *internet* seja a mesma do desenvolvimento para *Windows*.

Bragagnolo (2004) explica que o *ASP.NET* simplesmente gera páginas "HTML padrão", compatíveis com todos os navegadores, facilitando a interface com o usuário. Microsoft (2004) expõe casos de sucesso de empresas que utilizaram a tecnologia *.NET* na construção de sistemas. Um dos exemplos é o Banco Bradesco, que utiliza a tecnologia nas plataformas de negócios Bradesco Empresas e Bradesco Prime. Estas duas plataformas foram desenvolvidas para atender pessoas jurídicas e correntistas.

A tecnologia *.NET* permite o acesso aos sistemas e aos bancos de dados do Bradesco diretamente de suas estações de trabalho, e também disponibiliza produtos e serviços rapidamente para os clientes.

4 DESENVOLVIMENTO DO TRABALHO

O presente trabalho resultou na criação de um componente *Web* que provê a troca de informações entre dois sistemas distintos utilizando a tecnologia *Web Services* através da plataforma *.NET*. Validando o estudo desta tecnologia, utiliza-se um ambiente caracterizado B2B para que haja a troca de informações.

Nos próximos tópicos serão detalhados a especificação e os requisitos do sistema proposto para a troca de informações e as aplicações que utilizam este recurso.

4.1 ESPECIFICAÇÃO

Neste tópico serão mostradas as especificações da aplicação para a troca de informações via *internet*.

O mecanismo de troca de informações desenvolvido neste trabalho busca atender a necessidade de duas entidades conhecerem-se e trocarem informações entre elas, de forma que todos os dados sejam trafegados via *internet*. Ilustrando o funcionamento, os aplicativos desenvolvidos para demonstrar a troca de informações são especificados nos casos de uso onde foi implementado cada um deles para funcionar em diferentes plataformas, a fim de demonstrar funcionalidade entre diversos tipos de aplicação.

A aplicação principal trata-se do *Web Service* que será responsável pela troca de informações. Ela funciona como um repositório de regras e dados, e seus recursos tornam-se públicos para que as aplicações possam acessá-las. O armazenamento dos dados e todas as funcionalidades e requisitos especificados neste trabalho, foram implementados neste componente.

Deste modo, as aplicações para demonstração não funcionam como sistemas complexos, ou repositório de dados, e sim como interfaces para que o usuário entenda como os dados possam ser transitados e trabalhados de várias formas pelas diversas interfaces.

A especificação desta aplicação foi baseada na Linguagem de Modelagem Unificada (UML) baseado em Furlan (1998), e a ferramenta utilizada para elaboração dos diagramas foi o *Microsoft Visio*. Esta ferramenta possui suporte ao desenvolvimento de todos os diagramas da UML contemplados neste trabalho, e também acompanha a

versão *Architect* do *Visual Studio .NET*. Os diagramas especificados foram: diagrama de casos de uso, diagrama de classes, e diagrama de seqüência.

O tópico seguinte descreve os requisitos funcionais e requisitos não funcionais do desenvolvimento do sistema proposto.

4.1.1 Requisitos

O quadro 8 apresenta os requisitos funcionais previstos para o sistema de *Web Services.* Para a interface do sistema hotel e do sistema da agências de turismo, os requisitos especificados foram implementados na solução.

Requisitos Funcionais
RF01: *Web Service*: Cadastrar operadoras.
RF02: *Web Service*: Cadastrar hotéis.
RF03: *Web Service*: Cadastrar elementos de locação.
RF04: *Web Service*: Efetuar reserva.
RF05: *Web Service*: Efetuar locação.
RF06: *Web Service*: Buscar informações de reservas.
RF07: *Web Service*: Buscar informações do hotel.
RF08: *Web Service*: Efetuar locação.
RF09: Sistema Hotel: Cadastrar elementos de locação.
RF10: Sistema Hotel: Buscar informações de reservas.
RF11: Sistema Operadora: Efetuar reserva.
RF12: Sistema Operadora: Buscar informações de hotéis.

Quadro 8: Requisitos funcionais.

O quadro 9 lista os requisitos não funcionais previstos para o sistema, identificando os requisitos que serão contemplados na implementação.

Requisitos Não Funcionais
RNF01: Utilização do protocolo SOAP para troca de informações.
RNF02: Interface pública compartilhada.

RNF03: Segurança com senha para as entidades.
RNF04: Independência de plataforma.

Quadro 9: Requisitos não funcionais.

A aplicação *Web Service* faz com que os serviços fiquem disponíveis na *internet* em um servidor HTTP. Sua interface é pública, sendo assim suas funcionalidades podem ser acessadas por qualquer entidade com acesso ao servidor. Para que uma aplicação possa consumir este *Web Service*, basta ter acesso ao servidor e implementar o protocolo SOAP.

Pode-se criar uma aplicação *Web* ou uma aplicação *desktop* a partir de objetos e componentes disponíveis na *Web* (*Web Services*), já que o protocolo SOAP padroniza as mensagens trocadas entre as aplicações, tornando assim uma linguagem universal entre os *Web Services*.

O protocolo SOAP possibilita a interconexão entre os componentes envolvidos para haver a comunicação. O desenvolvimento do *Web Service* e os aplicativos para demonstração de troca de dados especificados neste trabalho, demonstram a integração entre aplicativos, mesmo que estes estejam em plataformas diferentes como uma aplicação *Web* e uma aplicação *desktop*. O protocolo HTTP facilita o transporte de mensagens SOAP entre os sistemas, uma vez que os *firewalls* normalmente não bloqueiam o acesso à porta HTTP.

4.2 DIAGRAMAS

Nesta seção estão especificados os diagramas utilizados neste trabalho, tais como diagrama de casos de uso, diagrama de classes e diagrama de seqüência. O item a seguir descreve o digrama de classes.

4.2.1 Digrama de Classes

O componente *Web Service* é um centralizador de informações, responsável pela troca de dados, regras e relacionamentos. Desta forma está sendo construído um componente capaz de armazenar e buscar dados, sendo que suas funcionalidades são acessadas por programas externos.

A descrição referente à implementação dos acessos ao componente *Web*, está descrito na seção que descreve a implementação.

Para especificar as regras de negócio da aplicação o digrama da figura 6 ilustra o diagrama de classes do *Web Service*. Este diagrama permite um melhor entendimento do funcionamento interno do componente *Web Service* implementado neste trabalho.

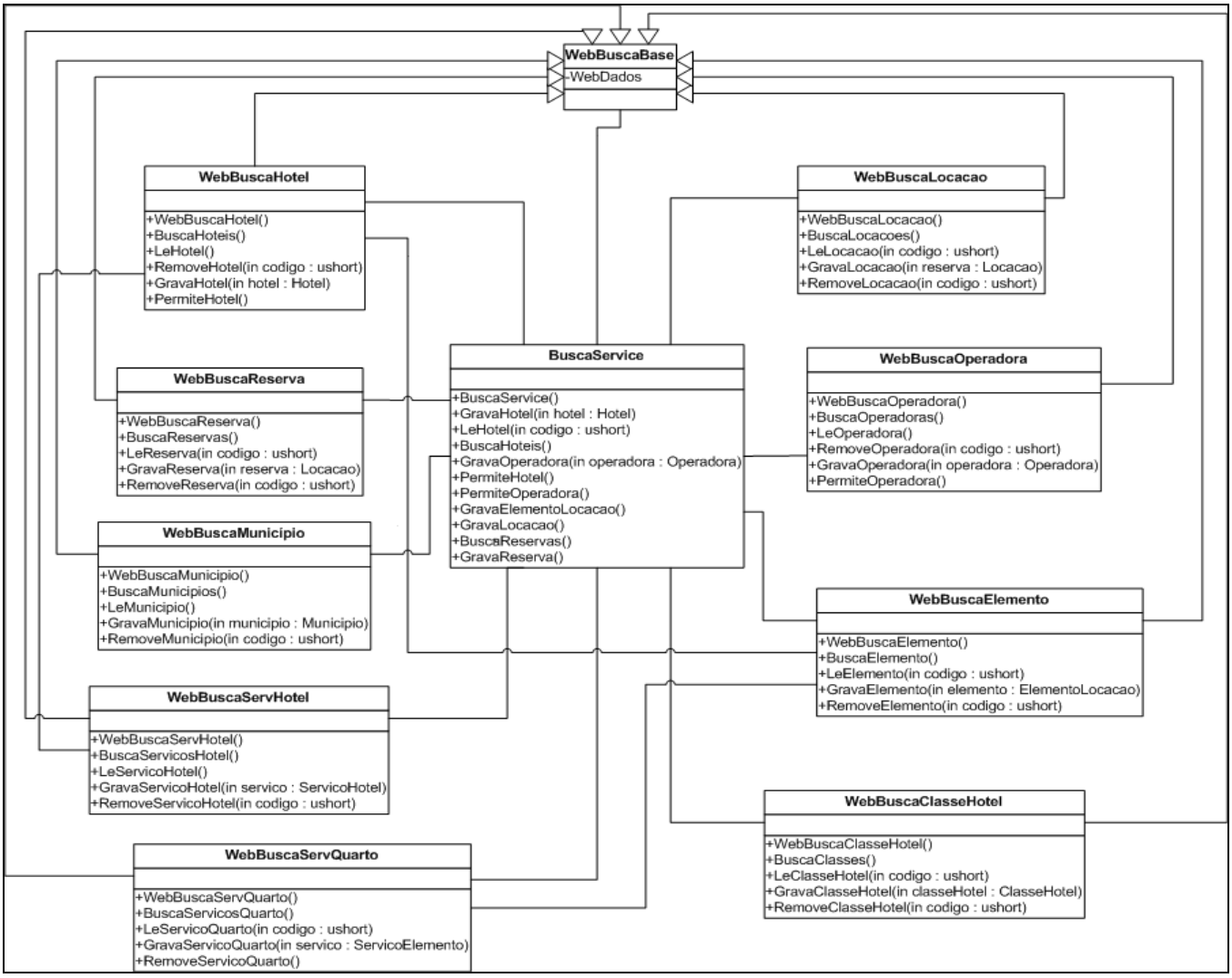


Figura 6: Diagrama de classes da aplicação.

A classe *BuscaService*, implementa o *Web Service* propriamente dito, sendo assim, funciona como uma interface pública para as demais aplicações. É nesta classe que estão implementados os métodos públicos, métodos estes que carregam o prefixo *WebMethod* para identificar um método do *Web Service*.

Para manipulação dos dados, foram criadas classes específicas de manutenção para as diversas tabelas, estas invocadas pela classe *BuscaService*. Todas derivam da

classe *WebBuscaBase*, já que todos possuem um objeto *DataSet* de manipulação dos dados em memória.

A seguir a descrição de cada uma das classes do *Web Service*:

WebBuscaHotel: Esta classe é responsável pela manutenção da tabela HOTEL, ela trata os objetos Hotel.

WebBuscaOperadora: Esta classe é responsável pela manutenção da tabela OPERADORA, ela trata objetos Operadora.

WebBuscaMunicipio: Esta classe implementa as rotinas de manutenção da tabela MUNICIPIO, ela trabalha com os objetos Municipio.

WebBuscaClasseHotel: Esta classe implementa as rotinas de manutenção da tabela CLASSEHOTEL, utiliza a classe ClasseHotel como parâmetro de manutenção.

WebBuscaReserva: Esta classe implementa as rotinas de manutenção da tabela LOCACAO, quando esta tratar de reservas, trata objetos Locacao.

WebBuscaServHotel: Responsável por tratar os dados da tabela de "SERVICOHOTEL" e "HOTEL_SERVICOHOTEL", utilizando a classe ServicoHotel

WebBuscaServQuarto: Responsável por tratar os dados da tabela SERVICOSQUARTO e ELEMENTO_SERVICOSQUARTO, utiliza a classe ServicoElemento.

WebBuscaElemento: Responsável por tratar os dados da tabela ELEMENTO, trabalha com objetos Elemento.

WebBuscaLocacao: Responsável por tratar os dados da tabela LOCACAO, trabalha com objetos Locacao.

Visando organização, padronização e segurança, os dados estão organizados em classes, desta forma toda informação trafegada entre a aplicação e o *Web Service* estarão contidos em objetos.

O diagrama da figura 7 ilustra o diagrama de classes de dados da aplicação, isto é, as classes representadas neste diagrama são responsáveis pela persistência dos dados no Banco de Dados. Com este diagrama pode-se entender quais serão os tipos válidos de entrada e saída de dados da aplicação

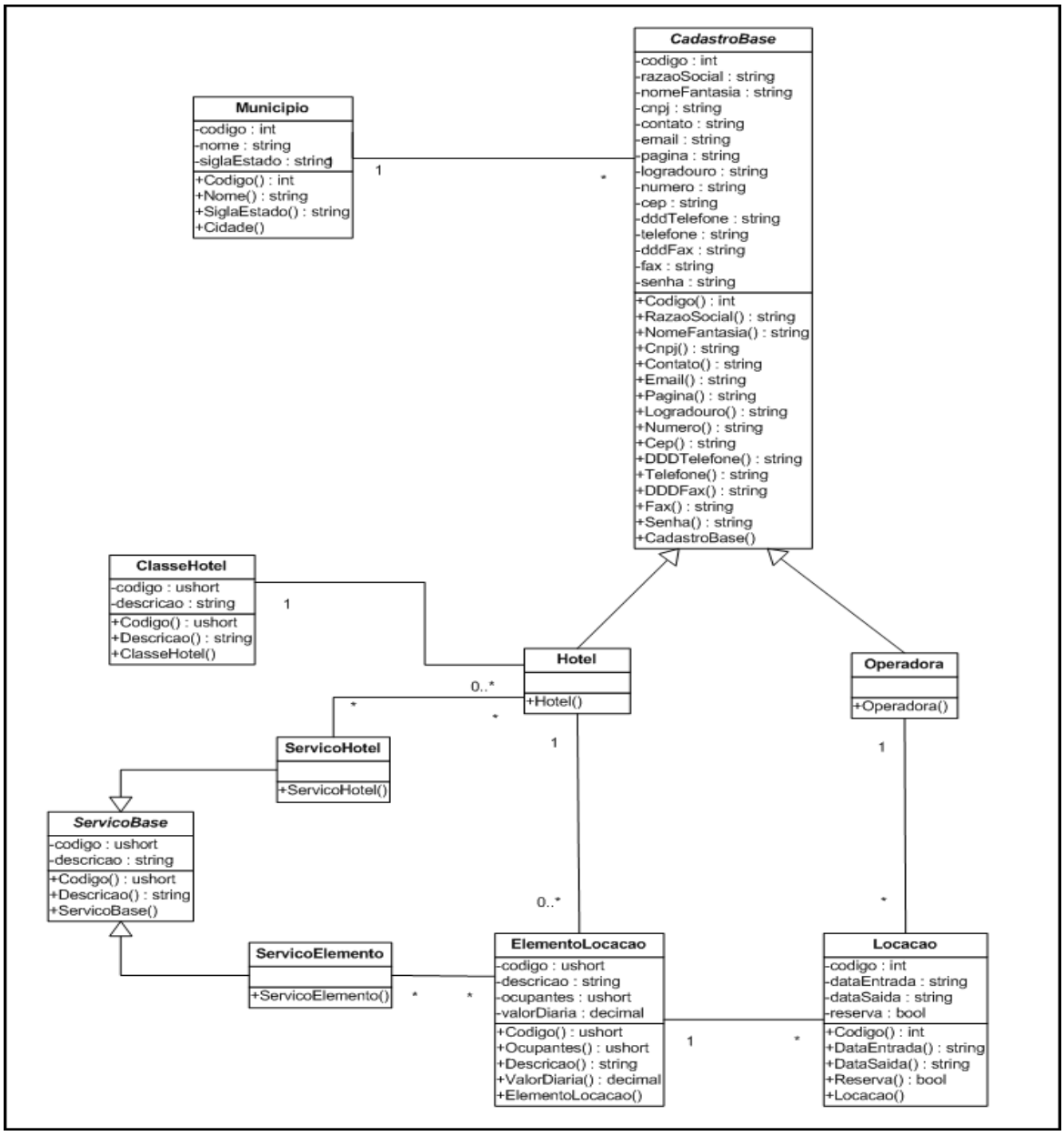


Figura 7: Diagrama de classes dos dados.

Cada uma das classes possui seus atributos privados que são acessados somente dentro das mesmas, estes atributos estão descritos com a inicial minúscula no diagrama de classes. A classe também possui propriedades públicas para serem acessadas a partir de outras classes, estas propriedades estão descritas com a inicial maiúscula e já é implícito a implementação da função *get* e *set* para atribuir e capturar os dados.

As funcionalidades de cada uma das classes estão descritas a seguir.

a) Município: esta classe define o município que será informado para as entidades, sendo esta informação obrigatória para ambas às entidades. Esta tabela será fixa no banco de dados para efeito de integridade.

b) CadastroBase: define o modelo para a construção das classes descendentes que possuem dados cadastrais.

c) Operadora: esta classe existe para definir as agências de turismo, e é descendente da classe CadastroBase.

d) Hotel: esta classe existe para definir os hotéis, sendo esta descendente da classe CadastroBase.

e) ClasseHotel: esta classe serve para definir em que classe o hotel pertence, 3 estrelas ou 5 estrelas, por exemplo. Esta tabela também será fixa no banco de dados para efeito de integridade.

f) ServicoBase: classe base para definir os serviços.

g) ServiçoHotel esta classe é derivada da classe ServiçoBase e serve para definir quais são os serviços que o Hotel possui. Por exemplo: piscina, sauna, quadra de esportes , sala de jogos, etc. Esta tabela será fixa no banco de dados para efeito de integridade.

h) ServicoElemento esta classe é derivada da classe ServiçoBase e serve para definir quais são os serviços que o elemento possui. Por exemplo: Televisão, ar-condicionado, frigo-bar, etc. Esta tabela será fixa no banco de dados para efeito de integridade.

i) ElementoLocacao: esta classe mantém um elemento possível de locação do hotel, por exemplo, o hotel deixará disponível para locação seus apartamentos, chalés, suítes entre outros.

j) Locacao: esta classe define a locação ou a reserva, quando for efetuada uma reserva pela operadora ou efetuada uma locação pelo hotel.

4.2.2 Diagrama de Casos de Uso

Para ser melhor compreendida a interação entre a aplicação principal *Web Service* e as demais aplicações desenvolvidas, foi elaborado o diagrama com os casos de uso, ilustrado na figura 8.

O sistema *Web Service* é visto como a caixa preta que fornece as situações da aplicação, de forma que é definido o comportamento sem revelar sua estrutura interna. As funcionalidades do sistema foram adicionadas neste contexto para fornecer uma descrição consistente e clara sobre as responsabilidades cumpridas pelo sistema.

O ator Entidade Hotel é responsável por criar uma conta de acesso para que o Sistema Hotel possa ter acesso ao *Web Service*. Assim como o ator Entidade Agência de turismo é responsável por criar uma conta de acesso para o Sistema Operadora.

O ator Sistema Hoteleiro é a interface do sistema Hotel, criada para demonstrar a troca de informações. O ator Sistema Operadora é a interface do sistema Operadora.

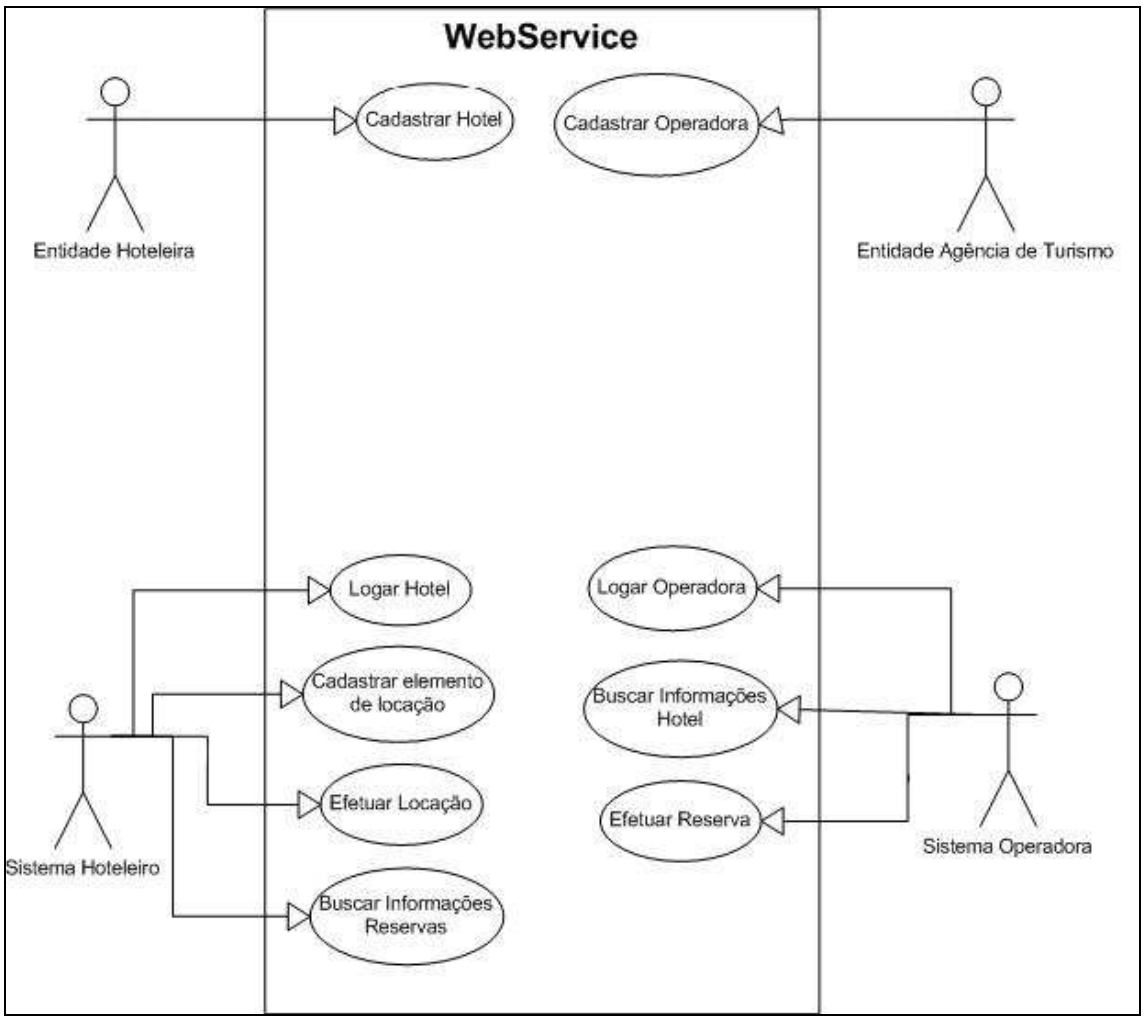


Figura 8: Caso de Uso *Web Service.*

Nos próximos itens serão descritos cada caso de uso com seu diagrama de seqüência.

O cadastro de entidades, representados nos casos de uso Cadastrar Hotel e Cadastrar Operadora é necessário para as entidades que desejam utilizar o *Web Service* possam cadastrar-se e tornarem-se clientes da aplicação. Só assim elas poderão ter acesso às demais funcionalidades do *Web Service*.

Para o cadastro, as entidades deverão enviar ao *Web Service* um cadastro simples da empresa que utilizará o recurso. São ilustrados nas figuras 9 e 10 os diagramas de seqüência referente aos casos de usos Cadastrar Hotel e Cadastrar Operadora respectivamente.

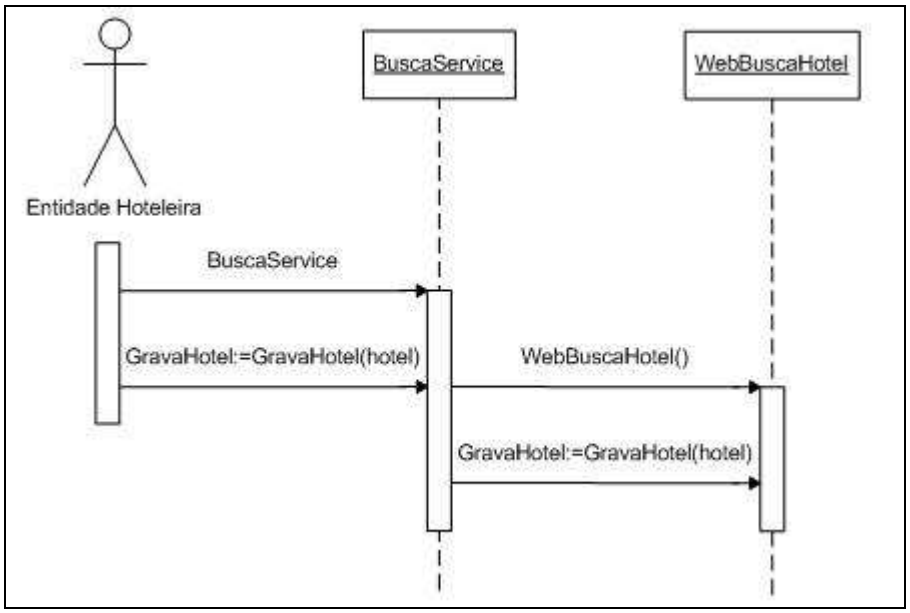


Figura 9: Diagrama de seqüência “Cadastrar Hotel”.

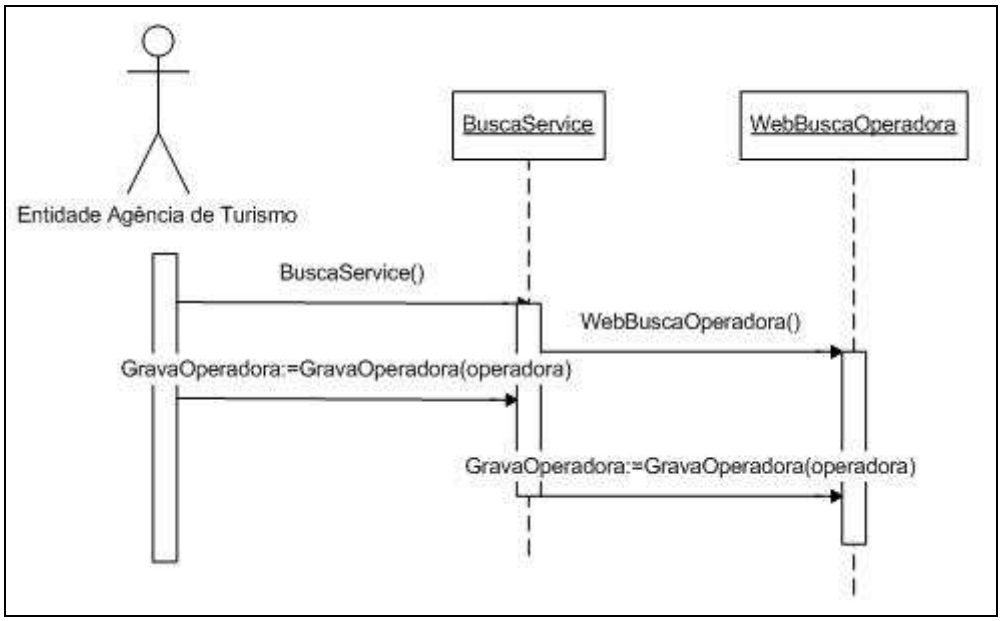


Figura 10: Diagrama de seqüência “Cadastrar Operadora”.

Antes de utilizar os recursos do *Web Service* os sistemas externos irão se logar no componente. O sistema hoteleiro e o sistema da operadora deverão implementar este procedimento pois só assim estarão aptos a usar as demais funções. Esta função está definida no caso de uso Logar Hotel e Logar Operadora.

Para melhor descrever a seqüência, foi elaborado o diagrama de seqüência. São ilustrados nas figuras 11 e 12 os diagramas de seqüência referente aos casos de uso Logar Hotel e Logar Operadora.

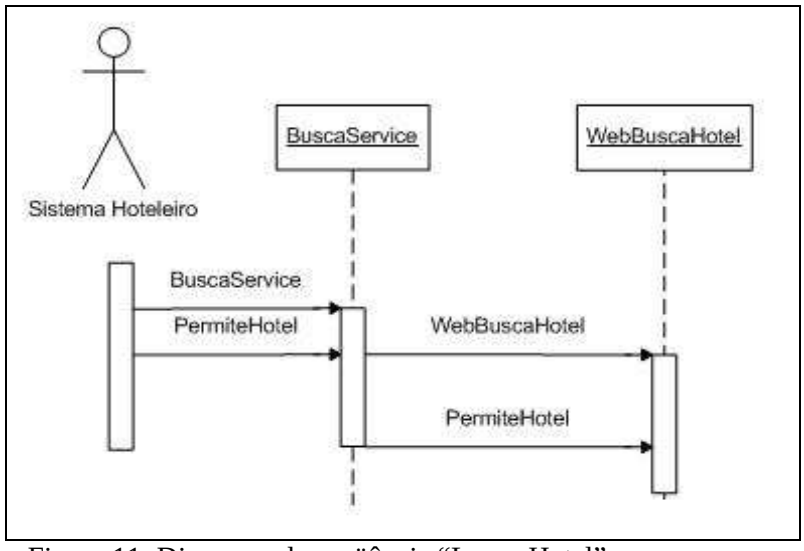


Figura 11: Diagrama de seqüência “Logar Hotel”.

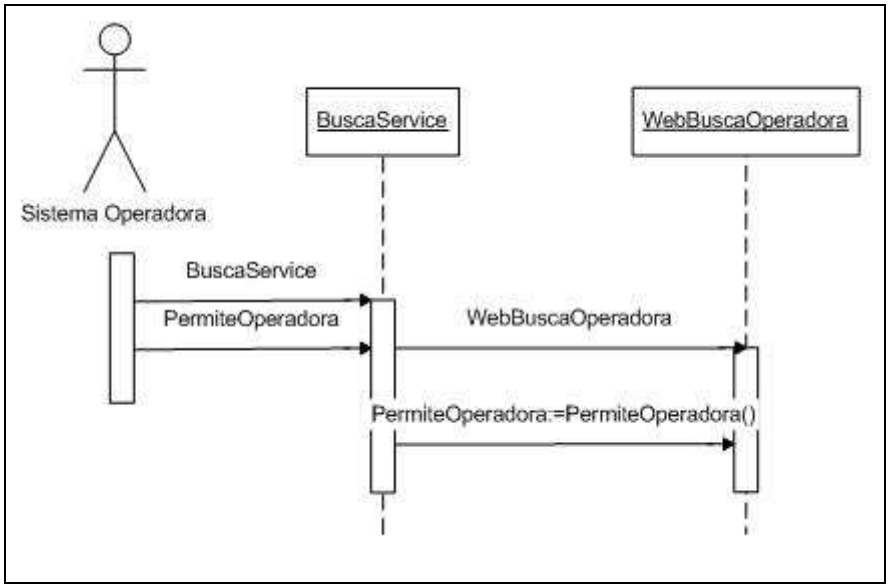


Figura 12: Diagrama de seqüência “Logar Operadora”.

O caso de uso Cadastrar Elemento de Locação descreve o cadastro de um elemento de locação efetuado pelo sistema hoteleiro. Os elementos de locação serão

armazenados como itens do objeto hotel, que servirá para a pesquisa de hotéis descritos nos próximos itens.

O caso de uso referente a este procedimento está implementado em “Cadastrar Elemento de Locação”. A figura 13 apresenta o diagrama de seqüência da interação entre as classes necessárias para efetuar esta ação.

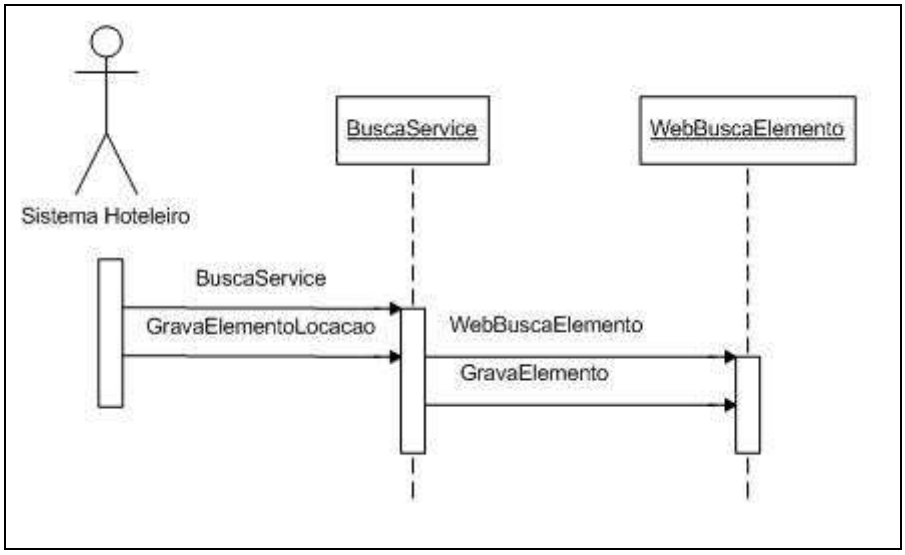


Figura 13: Diagrama de seqüência “Cadastrar Elemento de Locação”.

O caso de uso Efetuar Locação é necessário caso o hotel faça uma locação para um elemento, para que este elemento não fique disponível para reserva depois de locado. O caso de Uso “Efetuar Locação” descreve este procedimento. O diagrama de seqüência referente a esta função está descrito na figura 14.

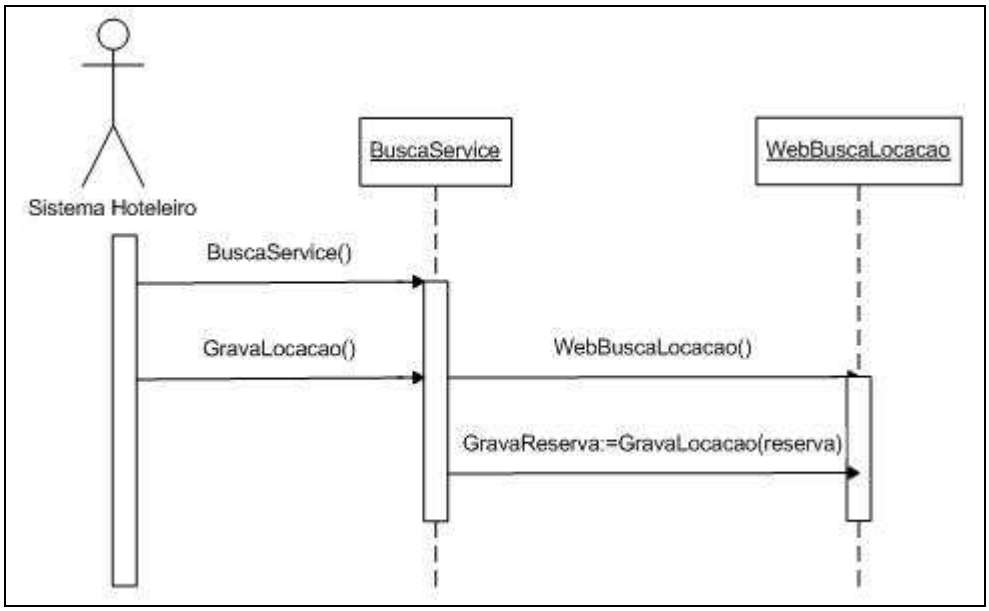


Figura 14: Diagrama de seqüência “Efetuar Locação”.

Esta função descrita no caso de uso Buscar Informações Reservas, é necessária para que os sistemas hoteleiros possam se atualizar em relação a reservas marcadas para determinados elementos de locação do hotel, oriundas das operadoras através do sistema *Web Service*. O diagrama de seqüência para esta função está especificado no diagrama de seqüência da figura 15.

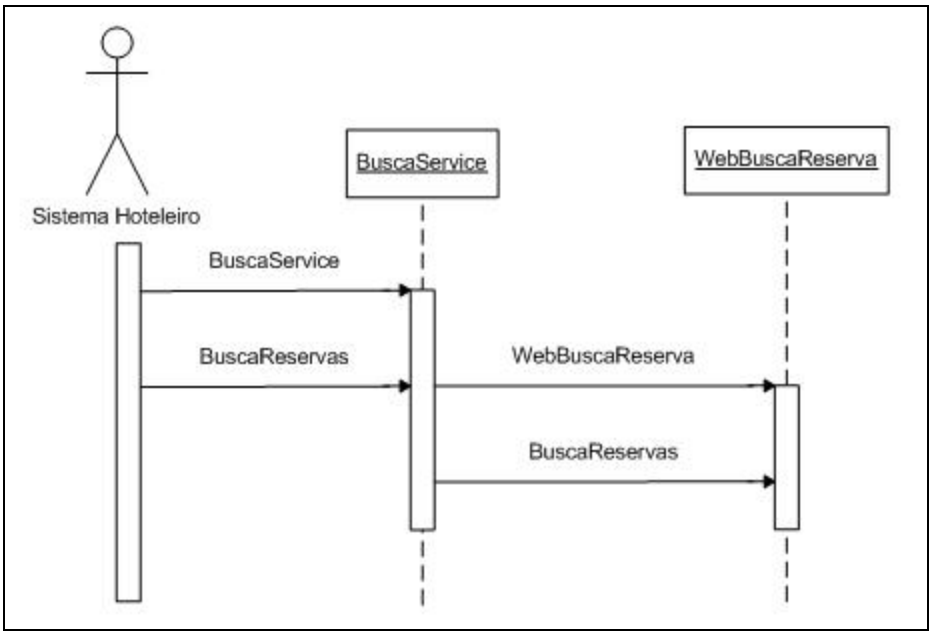


Figura 15: Diagrama de seqüência “Buscar Informações Reservas”.

Das funções de requisição de dados do *Web Service*, o caso de uso Buscar Informações Hotel é de extrema relevância. Esta talvez seja a mais utilizada. Buscar Informações Hotel é justamente a função que permite que as operadoras possam fazer as pesquisas de hotéis e elementos de locação, para uma posterior reserva. O Sistema Operadora entra com um filtro determinado, onde estarão informados os campos para poder buscar um hotel/elemento de acordo com o desejado. O sistema *Web Service* recebe este filtro e faz uma pesquisa no banco de dados para encontrar os hotéis e elementos que se encaixem no filtro.

Como esta função interage com várias tabelas, ela torna-se uma das funções mais complexas do *Web Service*. Inicialmente a função busca os hotéis e serviços, em

seguida busca os elementos de locações e serviços, isso se estiverem aplicados filtros para os elementos. Assim sendo, serão retornados todos os hotéis com seus devidos elementos de locação que se encaixem no filtro.

Para descrever esta função foi elaborado o diagrama de seqüência, o qual está ilustrado na figura 16.

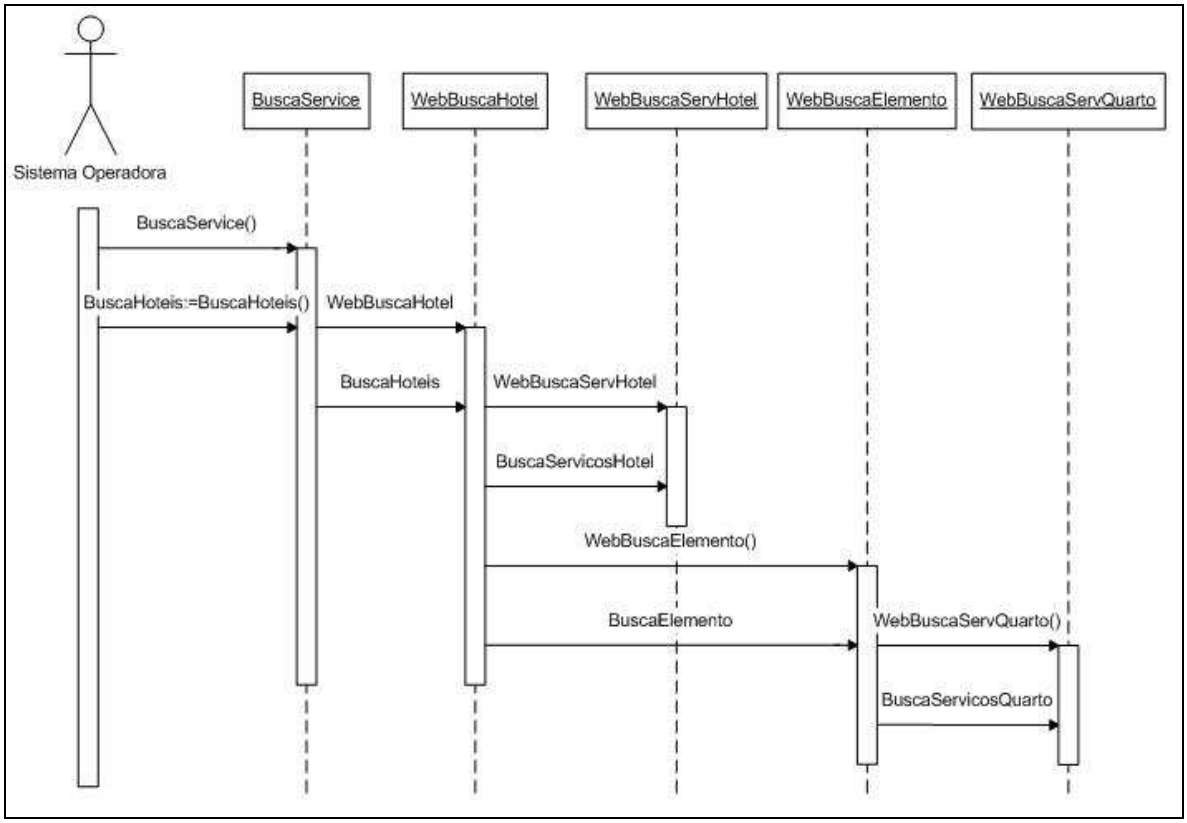


Figura 16: Diagrama de seqüência “Buscar Informações Hotel”.

O Sistema Operadora pode registrar uma reserva para um determinado elemento de locação. Esta reserva trata-se de um registro com a data de entrada e saída, qual operadora efetuou a reserva e o elemento de locação reservado. Para uma posterior pesquisa, este elemento não poderá ser reservado para este mesmo período. A entidade hotel deverá atualizar sua lista de reserva através da função do *Web Service*, para que um elemento não seja locado duas vezes.

Para descrever esta função a figura 17 ilustra o diagrama de seqüência deste caso de uso denominado Efetuar Reserva.

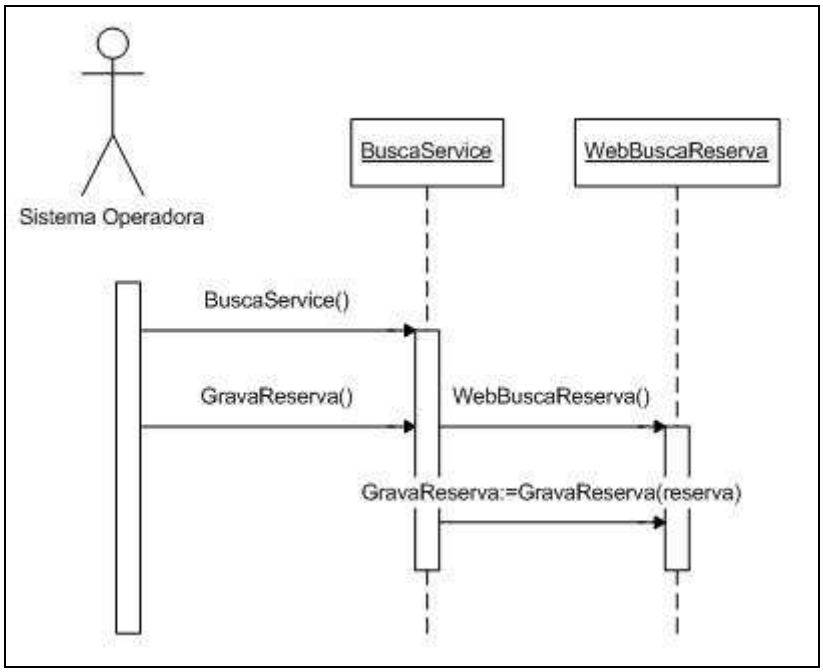


Figura 17: Diagrama de seqüência “Efetuar Reserva”.

4.2.3 Modelo de Entidade Relacionamento

A figura 18 apresenta o modelo físico para persistência dos dados através da utilização de um banco de dados. Este modelo é capaz de atender aos requisitos funcionais da aplicação.

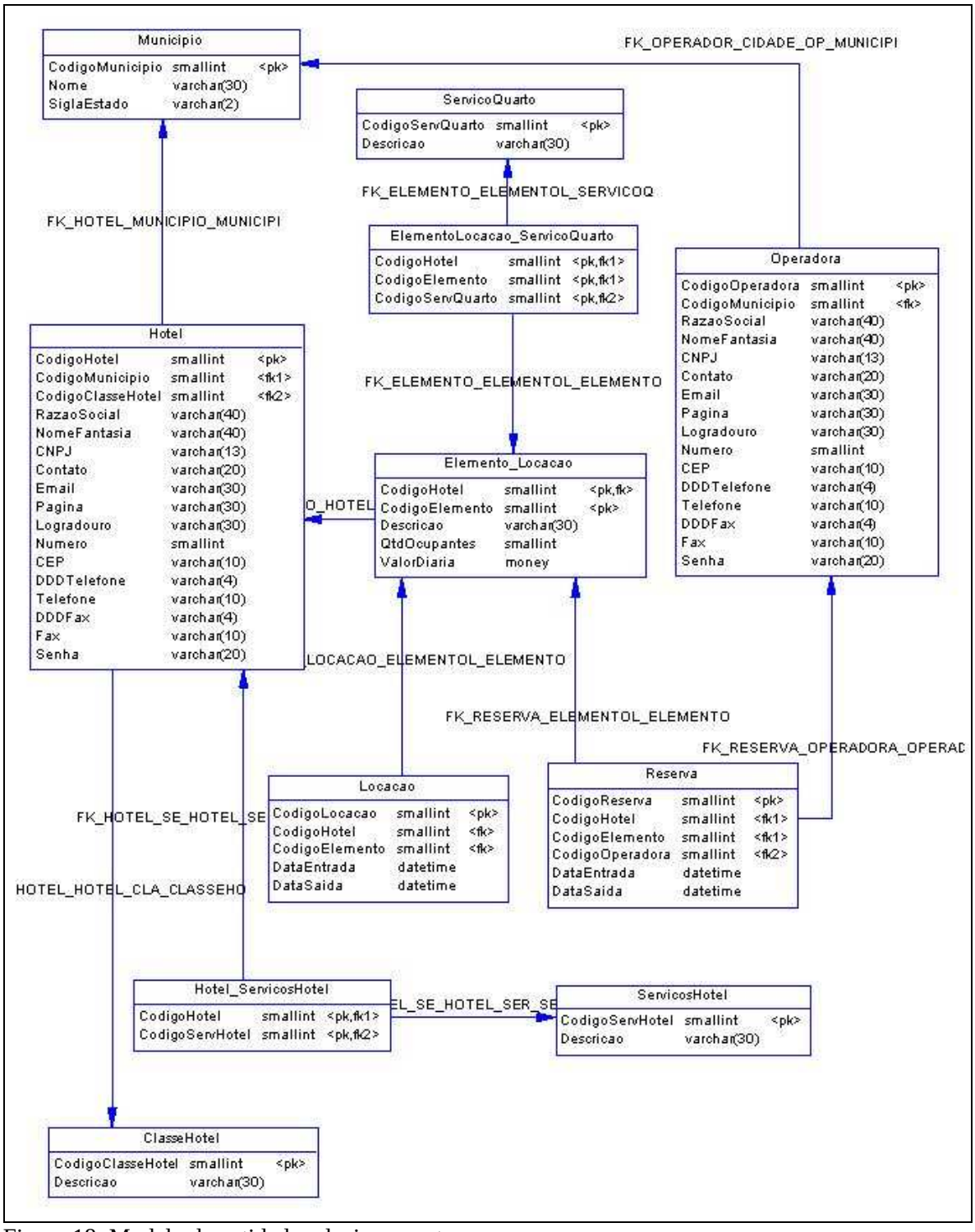


Figura 18: Modelo de entidade relacionamento.

4.3 IMPLEMENTAÇÃO

Nesta seção são apresentados os aspectos técnicos relacionados ao processo de implementação da solução para a troca de informações. Além da aplicação principal

desenvolvida, descreve-se a implementação dos outros sistemas clientes que fazem a interação com o componente principal.

4.3.1 Técnicas e Ferramentas Utilizadas

Foi escolhida para o desenvolvimento da solução de integração de Sistemas B2B a plataforma *.NET*. Como ferramenta de desenvolvimento foi utilizado o *Visual Studio .NET*, que possui facilitadores importantes no desenvolvimento *Web*, visto que esta traz um novo conceito de desenvolvimento de *software* voltado para a *internet*. No desenvolvimento das aplicações foi utilizada a linguagem C#.

Para o desenvolvimento da aplicação principal, foi utilizado o conceito de *Web Services*, que implementa interfaces acessíveis via SOAP. O banco de dados utilizado para armazenar os dados desta aplicação é o *SQL Server 2000* da Microsoft, que possui total integração a ferramenta *Visual Studio .NET*. A utilização do banco foi com base nas informações fornecidas por Microsoft (2002).

As aplicações clientes foram desenvolvidas para o acesso ao *Web Service* e obrigatoriamente implementam o protocolo SOAP. Para demonstrar os vários meios de acesso ao *Web Service* foram desenvolvidos dois clientes distintos que funcionam em diferentes plataformas.

Visando atender o conceito empregado pela Microsoft de desenvolvimento em camadas, foi criado um projeto que abrange somente as classes de dados utilizados pelo *Web Service*. Este projeto serve como referência para objetos trafegados via SOAP nos parâmetros de entrada e saída das funções do *Web Service*.

Para o desenvolvimento de um dos clientes foi utilizado o modelo de aplicação *WindowsForms*, uma aplicação em formato janela. Outra aplicação cliente baseada no modelo de aplicação *WebForms*, desenvolvido como uma aplicação *ASP. NET*, possui o código interpretado pelo servidor de aplicações ASP .NET a partir do *Internet Information Service* que gera uma interface no navegador HTML.

A figura 19 ilustra a proposta de funcionamento do mecanismo de comunicações entre o *Web Service*, Sistema Hoteleiro e Operadora (Agências de turismo)

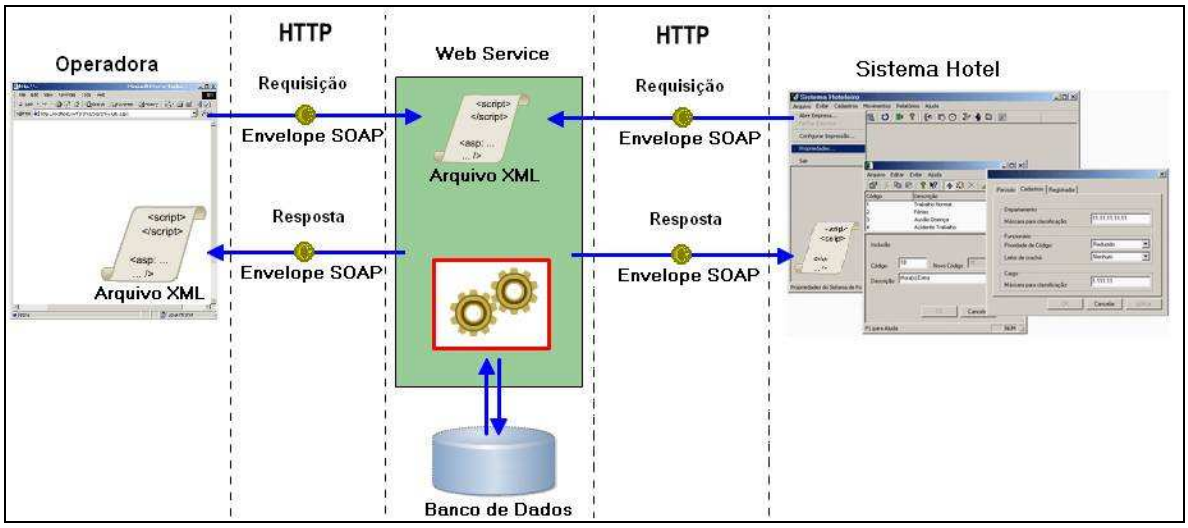


Figura 19: Esquema de comunicação do *Web Service*.

4.3.2 Aplicação *Web Service*

No desenvolvimento do componente *Web Service* foi utilizado os recursos da ferramenta *Visual Studio .NET*. Este componente funciona com um componente *Web* e é basicamente um repositório de dados. Neste componente foram implementadas as funcionalidades da integração entre os sistemas. Dentro da ferramenta, não há necessidade de se aprender um novo método de desenvolvimento para *Web Services*, apenas são adicionados nos objetos que se deseja tornar disponível na *Web* algumas atributos. Para o funcionamento correto do componente, deve ser montada uma infra-estrutura para que aceite as requisições de aplicações externas que chegam por meio do HTTP.

Para se criar um *Web Service* no *Visual Studio .NET*, basta criar um novo projeto e dizer que este é do tipo *ASP .NET Web Service*, como indicado na figura 20.

Figura 20: Criação do *Web Service*.

Depois de criar o projeto para o *Web Service*, é necessário implementar a interface gerada pela ferramenta para que as outras aplicações possam acessa-lo via SOAP. Esta interface é mantida pela classe comum principal derivada da classe *Web Service*. No presente trabalho apenas uma classe de interface foi implementada. Com o atributo "*WebDescription*" pode se colocar uma descrição para o *Web Service*, configurar a *namespace* e outras opções, como localização entre outros.

Nesta classe que são implementados os métodos de acesso ao componente, contudo, é necessário que seja incluído o atributo "*WebMethod*" para que os métodos tornem-se públicos.

O quadro 10 mostra parte do código fonte para demonstrar a declaração da classe *BuscaService* que funciona como a interface do *Web Service*. O quadro demonstra também como são declarados os métodos públicos.

```
using System;
using System.Collections;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Web;
using System.Web.Services;
using System.Data.SqlClient;
using Objetos;

namespace WebBusca
{
   /// <summary>
   /// Classe BuscaService, este serviço tem como finanlidade dar manutenção ao
   /// banco de dados de serviços de localização de Hotéis e efetuar reservas.
   /// </summary>
   [WebService(Description="Serviço de integração para Agências de turismo e hoteis.")]
   public class BuscaService : WebService
   {
       Atributos
       Propriedades

       #region WebMethods Hotel

       [WebMethod(Description = "Busca hotéis de acordo com o filtro")]
       public Hotel[] BuscaHoteis(FiltroHotel filtro)
       {
           return BuscaHotel.BuscaHoteis(filtro);
       }

       [WebMethod(Description = "Busca todos os hotéis")]
       public Hotel[] BuscaTodosHoteis()
       {
           return BuscaHotel.BuscaHoteis();
       }
```

Quadro 10: Declaração da classe *BuscaService.*

Depois de compilado, é gerado o programa *Web Service* que é capaz de receber requisições externas através da *internet*. Neste momento todos os clientes que implementem o protocolo SOAP podem fazer estas requisições.

Isto é possível através do documento WSDL gerado automaticamente após a compilação pelo *Visual Studio .NET*. O WSDL é responsável pela validação da chamada dos métodos, descrevendo o serviço do *Web Service*.

O *Visual Studio .NET* possui um recurso de visualização do documento WSDL através de uma página HTML tornando mais interessante e fácil à visualização. Para acesso a este arquivo basta executar a aplicação *Web Service*, que automaticamente será aberto um navegador com a página carregada, ou pode ir-se diretamente no navegador e colocar o caminho da aplicação, que no caso é : http://localhost/WebBusca/BuscaService.asmx.

A figura 21 demonstra o documento HTML gerado do *Web Service* do programa principal.

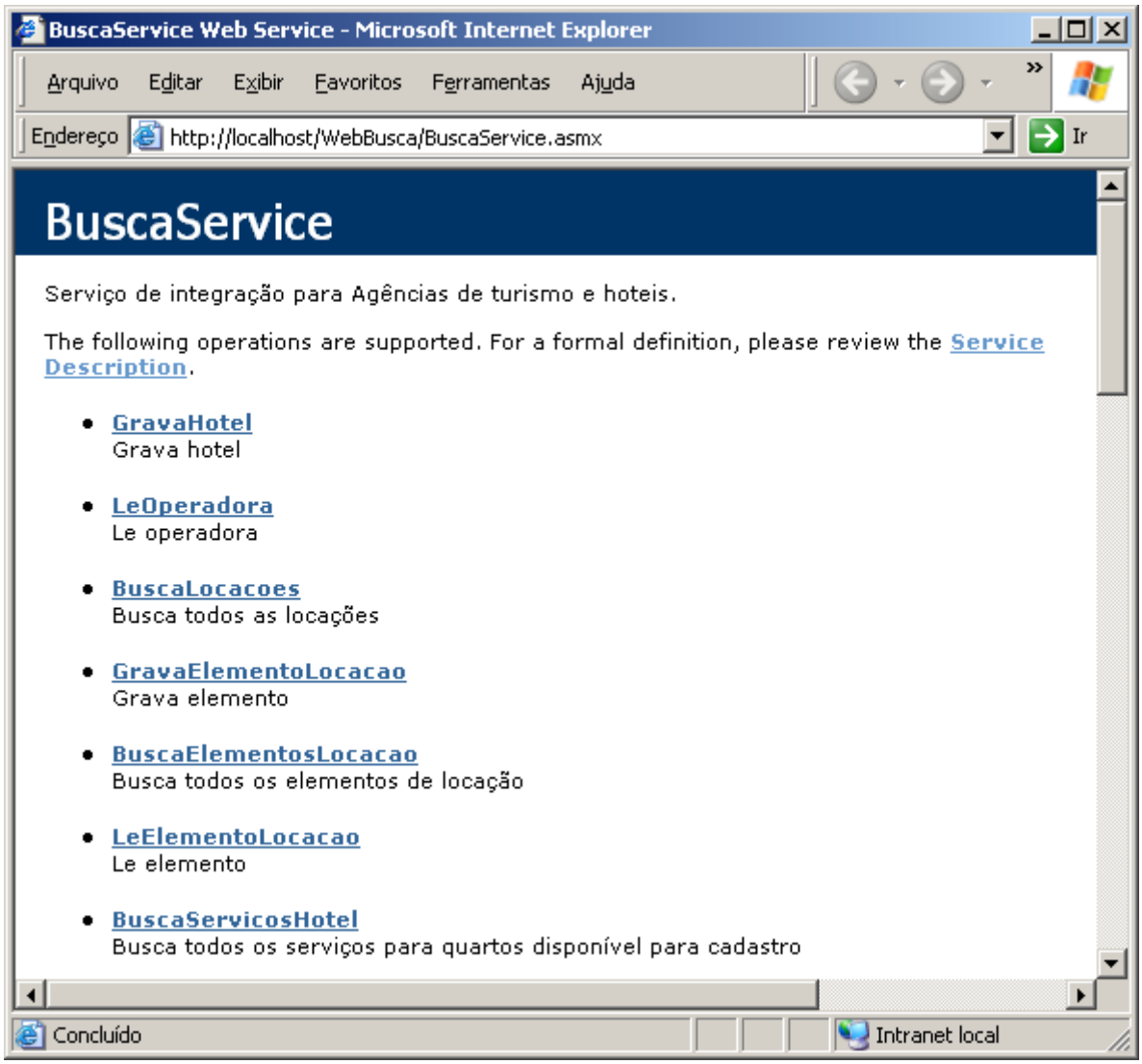


Figura 21: Documento HTML gerado pelo *Visual Studio .NET*.

A versão original do documento WSDL pode ser conseguida adicionando o sufixo “?WSDL” após o endereço da aplicação. No programa desenvolvido apresentar-se-á desta forma: http://localhost/WebBusca/BuscaService.asmx?WSDL.

Este documento em formato XML possui todos os detalhes que uma aplicação cliente necessite para poder acessar o *Web Service*. Estes detalhes são informações relevantes, como descrição, localização, formatação e requerimentos de interação.

A figura 22 representa uma parte do documento WSDL gerado pelo aplicativo *Web Service*.

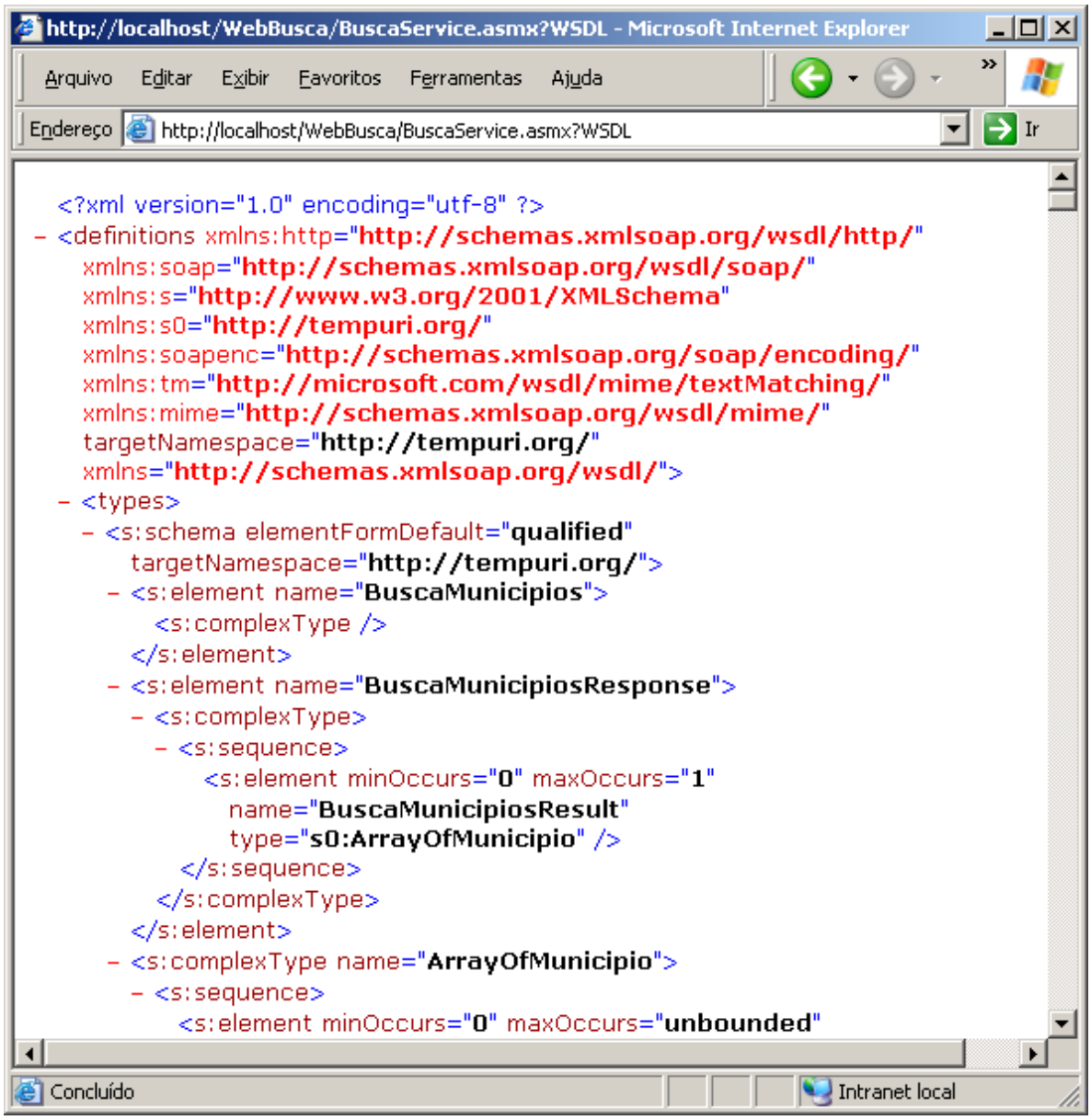


Figura 22: Documento WSDL gerado pelo *Web Service.*

4.3.3 Aplicações Clientes

A próxima etapa no desenvolvimento da solução, é criar as aplicações clientes necessárias para que os dados possam ser manipulados e visualizados externamente. O objetivo desta implementação é tornar a par o funcionamento da troca de informações que ocorrerá por intermédio do sistema principal *Web Service,* já especificado anteriormente.

Os clientes foram desenvolvidos em duas plataformas de execução diferentes. A aplicação que simula um sistema hoteleiro foi desenvolvida no padrão *Windows Forms* (aplicação windows comum). Para simular o sistema de uma agência de turismo e para o cadastro das entidades, o aplicativo foi desenvolvido no padrão *Web Forms* (página web).

Para as aplicações clientes, foram implementados os mínimos dos requisitos especificados nos casos de uso, apenas para demonstrar a funcionalidade da troca de dados.

Para que as aplicações clientes possam acessar os recursos do programa principal, é necessário que estes consumam o *Web Service*. Este processo é semelhante a fazer uma referencia a uma dll.

Para adicionar uma referência ao *Web Service,* basta utilizar o recurso do *Visual Studio .NET* responsável por referências de componentes *Web.* A função coloca a referência do *Web Service* em "*WebReferences*". A figura 23 ilustra a tela de adição a uma referência para o *Web Service* implementado.

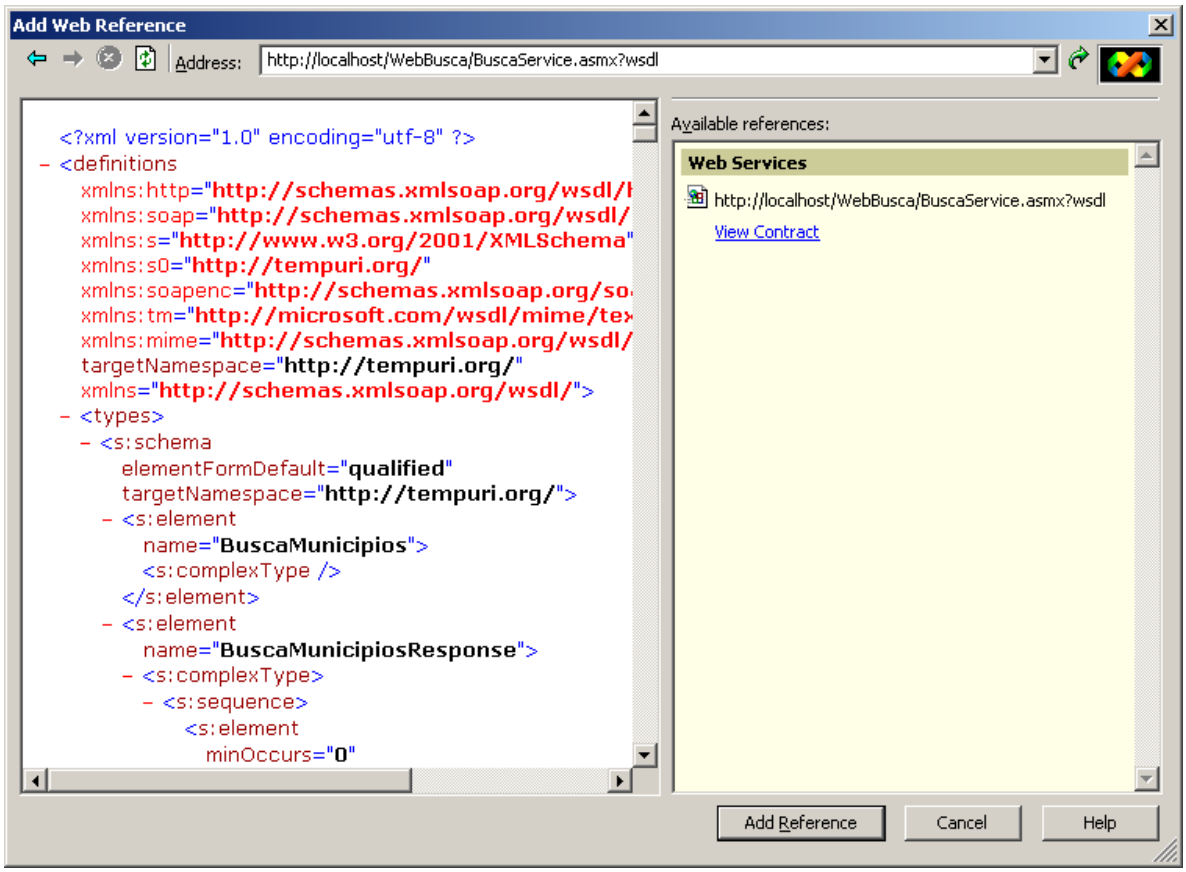


Figura 23: Adição de uma referência para o *Web Service*.

Depois de adicionado a referência para o *Web Service*, o *Visual Studio.NET* tem ferramentas que facilitam a utilização de *Web Services*, construindo um "*proxy*", por meio do qual é possível chamar qualquer *Web Service* como se fosse um objeto local.

Quando o cliente chama uma das funções de uma classe *proxy*, a classe cria uma solicitação HTTP e a envia ao servidor. Quando a resposta volta do servidor, a classe *proxy* analisa os resultados e os retorna à função que chamou. Isso permite uma interação uniforme entre clientes e servidores *Web* através de HTTP e XML.

4.3.3.1 Sistema Hotel

A aplicação do Sistema Hotel, simula um sistema utilizado por um Hotel para fazer seus controles básicos, como controle de quartos (elemento de locação), serviços, locações e reservas. Este sistema fará uma interconexão com o sistema *Web Service* para disponibilizar seus serviços como reservas de quartos e mesmo divulgação de serviços.

Este aplicativo desenvolvido funciona semelhantemente a um sistema integrado de um hotel, porém não foi desenvolvido para tal. Uma aplicação desenvolvida para este fim deve possuir seu próprio controle dos dados, como por exemplo uma conexão diretamente a um banco de dados. No programa desenvolvido. o aplicativo apenas trabalha com objetos localizados no servidor *Web* e a manipulação dos dados é feita través de requisições e respostas ao componente principal.

Para o aplicativo ter acesso à interface e objetos do componente principal, uma referência *Web* foi adicionada. A partir desta referência todos os recursos do componente *Web* foi disponibilizado bastando criar uma instância do objeto *Web*.

No quadro 11 há um exemplo de uma propriedade da classe principal que faz a criação do objeto da interface do componente *Web*.

```
localhost.BuscaService BuscaService
{
    get
    {
        if(buscaService == null)
            buscaService = new localhost.BuscaService();

        return buscaService;
    }
    set
    {
        buscaService = value;
    }
}
```

Quadro 11: Instância do componente *Web Service.*

Na figura anterior a referência para o objeto *Web* cria a classe proxy *localhost*, a partir desta é possível enxergar a classe de interface do *Web Service* denominada "*BuscaService*".

4.3.3.2 Sistema Operadora

Esta aplicação simula uma agência de turismo que possui sua própria página na *internet*. A interconexão ao *Web Service* irá possibilitar que esta agência de turismo utilize sua aplicação *Web* como uma ferramenta de pesquisa de hotéis, através de filtros

específicos, listagem de quartos, entre outros. Esta pesquisa é feita através do sistema principal já implementado, tendo ainda o recurso de efetuar reservas para cada elemento de locação.

Esta aplicação *Web*, como o sistema hotel, irá possuir acesso à interface e objetos do componente principal através da referência *Web*. A partir do objeto *Web* instanciado, pode criar-se os filtros, efetuar as diversas buscas de informações bem como o envio de dados como para a reserva, por exemplo.

Para demonstrar o acesso ao componente principal através do componente do Sistema Operadora desenvolvido em *ASP .NET* o quadro 12 mostra a função criada para efetuar uma pesquisa de hotéis a partir de um filtro que é preenchido em uma página *Web*.

```
private void btPesquisar_Click(object sender, System.EventArgs e)
{
    localhost.FiltroHotel filtro = new localhost.FiltroHotel();
    filtro.CodigoMunicipio  = Convert.ToUInt16(cbDestino.SelectedItem.Value);
    filtro.CodigoClasse     = Convert.ToUInt16(cbClasse.SelectedItem.Value);
    filtro.QtdOcupantes     = Convert.ToUInt16(cbHospedes.SelectedItem.Value);
    filtro.DataEntrada      = Convert.ToString(dataEntrada.SelectedDate);
    filtro.DataSaida        = Convert.ToString(dataSaida.SelectedDate);

    filtro.ServicosHotel = new localhost.ServicoHotel[lbServSelecionados.Items.Count];

    int i = 0;
    foreach(ListItem item in lbServSelecionados.Items)
    {
        localhost.ServicoHotel serv = new localhost.ServicoHotel();
        serv.Codigo     = Convert.ToUInt16(item.Value);
        serv.Descricao  = item.Text;
        filtro.ServicosHotel[i++] = serv;
    }

    Session["filtro"] = filtro;
    Response.Redirect("ResultadoForm.aspx");
}
```

Quadro 12: Função de pesquisa do aplicativo cliente.

A referência para o objeto *Web* é identificada como *localhost*, semelhante ao sistema hoteleiro. O quadro 13 demonstra a criação da instância de um objeto do tipo *FiltroHotel* pertencente à interface *"BuscaService"*. Criada uma instância deste objeto localmente, seu funcionamento é igual a qualquer outro objeto.

```
private void btPesquisar_Click(object sender, System.EventArgs e)
{
    localhost.FiltroHotel filtro = new localhost.FiltroHotel();
    filtro.CodigoMunicipio  = Convert.ToUInt16(cbDestino.SelectedItem.Value);
    filtro.CodigoClasse     = Convert.ToUInt16(cbClasse.SelectedItem.Value);
    filtro.QtdOcupantes     = Convert.ToUInt16(cbHospedes.SelectedItem.Value);
    filtro.DataEntrada      = Convert.ToString(dataEntrada.SelectedDate);
    filtro.DataSaida        = Convert.ToString(dataSaida.SelectedDate);

    filtro.ServicosHotel = new localhost.ServicoHotel[lbServSelecionados.Items.Count];

    int i = 0;
    foreach(ListItem item in lbServSelecionados.Items)
    {
        localhost.ServicoHotel serv = new localhost.ServicoHotel();
        serv.Codigo     = Convert.ToUInt16(item.Value);
        serv.Descricao  = item.Text;
        filtro.ServicosHotel[i++] = serv;
    }

    Session["filtro"] = filtro;
    Response.Redirect("ResultadoForm.aspx");
}
```

Quadro 13: Instância do objeto *FiltroHotel*

4.3.4 Operacionalidade da implementação

O processo de troca de informações especificado neste trabalho está dividido em três funcionalidades básicos:

a) cadastro das entidades envolvidas: é onde são cadastrados os hotéis e operadoras de turismo que utilizam o sistema de integração;
b) cadastro de elementos de locação: onde o hotel informa quais os elementos possui disponível para reservas;
c) pesquisa e reserva: A operadora de turismo faz a busca dos hotéis desejados e executa uma reserva para um determinado elemento de locação.

Para demonstrar a funcionalidade do aplicativo desenvolvido para este trabalho, serão cadastradas as entidades hotel e agência de turismo. Onde o hotel irá fazer o cadastro dos elementos de locação, afim de torná-los disponíveis para a reserva através do sistema de integração. A agência de turismo irá fazer uma busca para localizar um quarto e uma posterior reserva, de acordo com a sua disponibilidade. O hotel ainda poderá executar uma busca de reservas para seus elementos, para atualização do banco de dados.

Os cadastros e demais funcionamentos estão melhores descritos nos cenários apresentados a seguir.

O cenário “Cadastro de Entidades” envolve dois casos de uso distintos. Um é o cadastro de hotéis e outro é o cadastro de operadoras (já detalhado no item “Casos de Uso”). O cadastro ocorre quando uma nova entidade irá utilizar o serviço de integração.

Este cadastro é feito a partir de uma página *Web*, num aplicativo identificado como *WebManutencao*, desenvolvido especificamente para esta função. Esta página também faz o acesso a interface do programa principal para gravação e leitura das entidades. Na figura 24 será exibida a tela de cadastro de hotel.

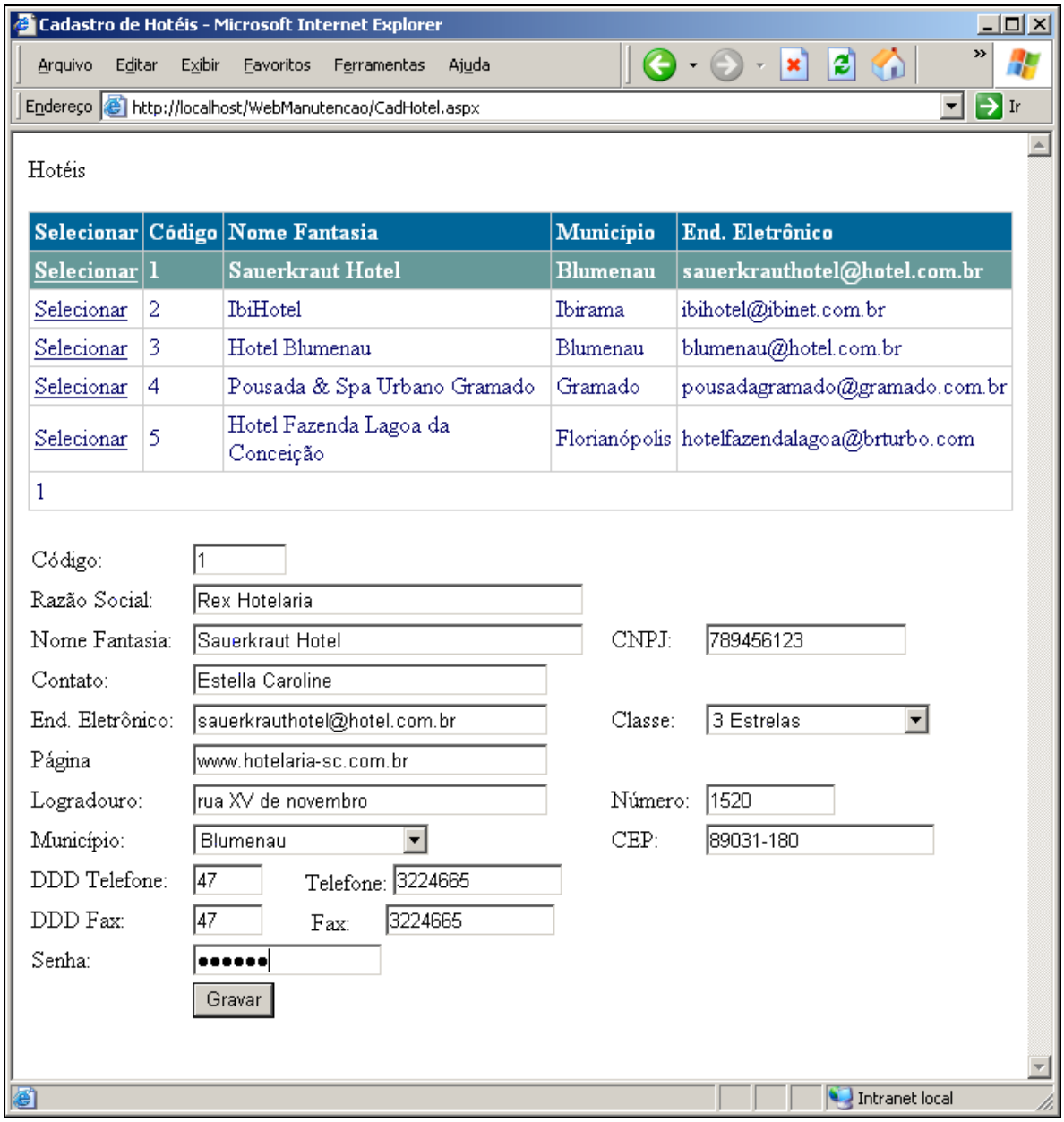


Figura 24: Cadastro da entidade Hotel.

Município e Classe do Hotel são campos previamente cadastrados pela *WebManutencao,* sendo que a entidade deverá optar por uma das opções existentes. Estes campos podem ser alterados no programa principal, uma vez que as funções para manipulação destas tabelas já se encontram implementados no programa principal. Contudo deve-se tomar cuidado com estas alterações, pois estão no nível de manutenção dos dados.

O restante do cadastro requer informações relevantes sobre a entidade, que devem ser preenchidas com as particularidades de cada qual. Ao ser cadastrado um hotel, todos os campos devem ser corretamente informados pois irá influenciar na integridade dos resultados obtidos nos filtros aplicados posteriormente pela interface da operadora.

O cadastro de operadoras é semelhante ao cadastro do hotel, todavia são abstraídos alguns campos apenas relacionados ao hotel, ilustrado na figura 25.

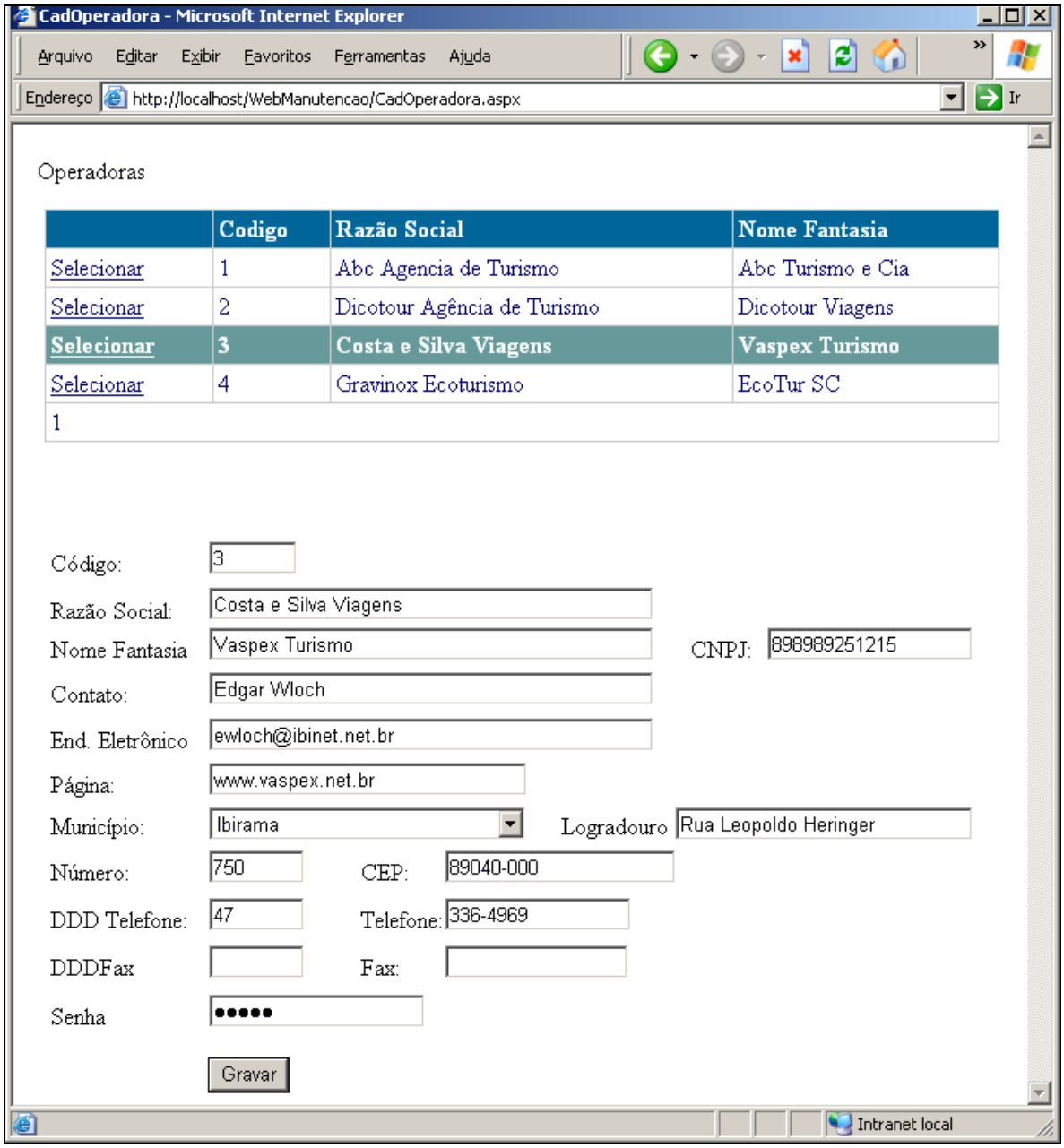


Figura 25: Cadastro da Entidade Operadora.

Os hotéis cadastrados informarão seus elementos de locação, e nesta seção são cadastrados os elementos disponíveis para locação, como quarto, apartamentos, chalés, etc. A figura 26 mostra a tela do sistema hotel, aplicativo desenvolvido no padrão de *Windows Forms*, que serve como cadastro dos elementos de locação. Nesta tela são informados os detalhes do elemento, como seus serviços, por exemplo. O correto cadastro dos elementos também influencia na integridade da pesquisa dos hotéis.

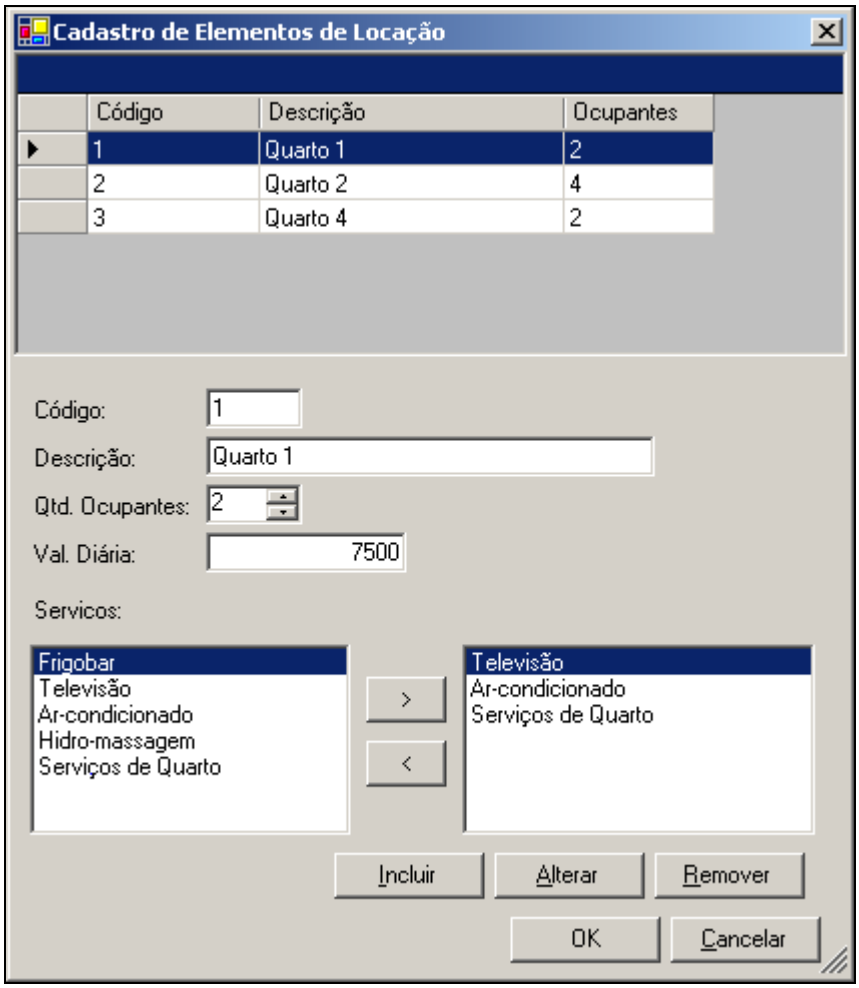


Figura 26: Cadastro do elemento de locação.

O cenário de pesquisa ou “Busca Hotéis” ocorre quando a interface da operadora executa uma procura para achar um determinado hotel, e reservar um quarto. Este cenário também está descrito nos casos de uso.

Esta tela de pesquisa também foi desenvolvida em *ASP .NET*, sendo executada em um navegador HTML. A figura 27 demonstra a tela de filtro implementada pelo sistema operadora.

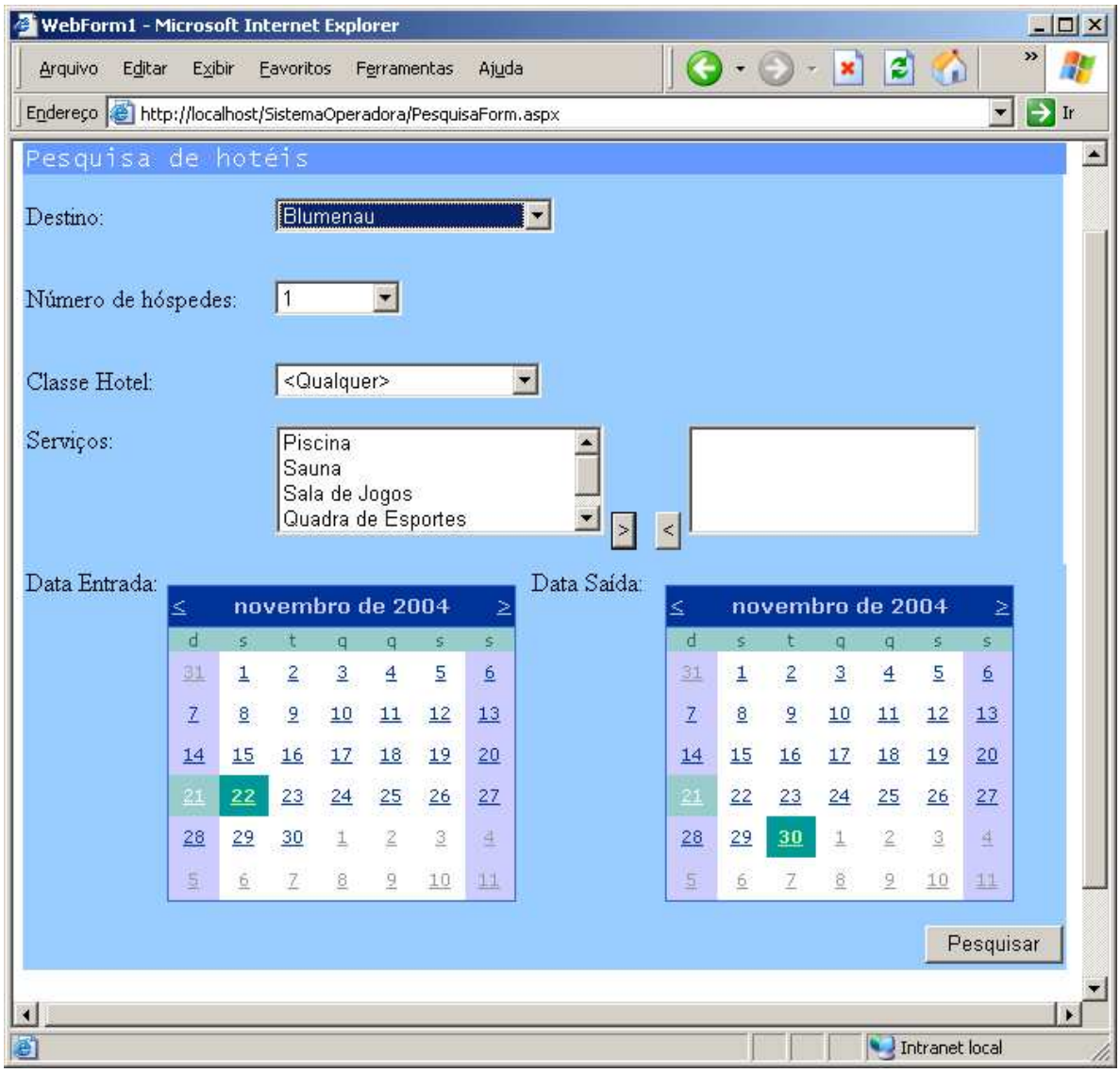


Figura 27: Tela de filtro para pesquisa de hotel.

Para o filtro foram apenas adicionados alguns campos para efeito de demonstração do recurso, contudo, no programa principal estão implementados demais filtros, como filtro para elementos de locação. Desta forma, basta apenas a inclusão dos campos em tela e enviar o filtro para o programa principal.

A partir do botão “pesquisar”, é feita uma requisição para o programa principal, enviando como parâmetro um objeto filtro, passando todas as propriedades desejadas. Em seguida a página *Web* é redirecionada para a página resultados, onde está exposto uma lista de hotéis que se encaixaram no filtro. A figura 28 demonstra uma lista de hotéis cadastrados que satisfazem o filtro.

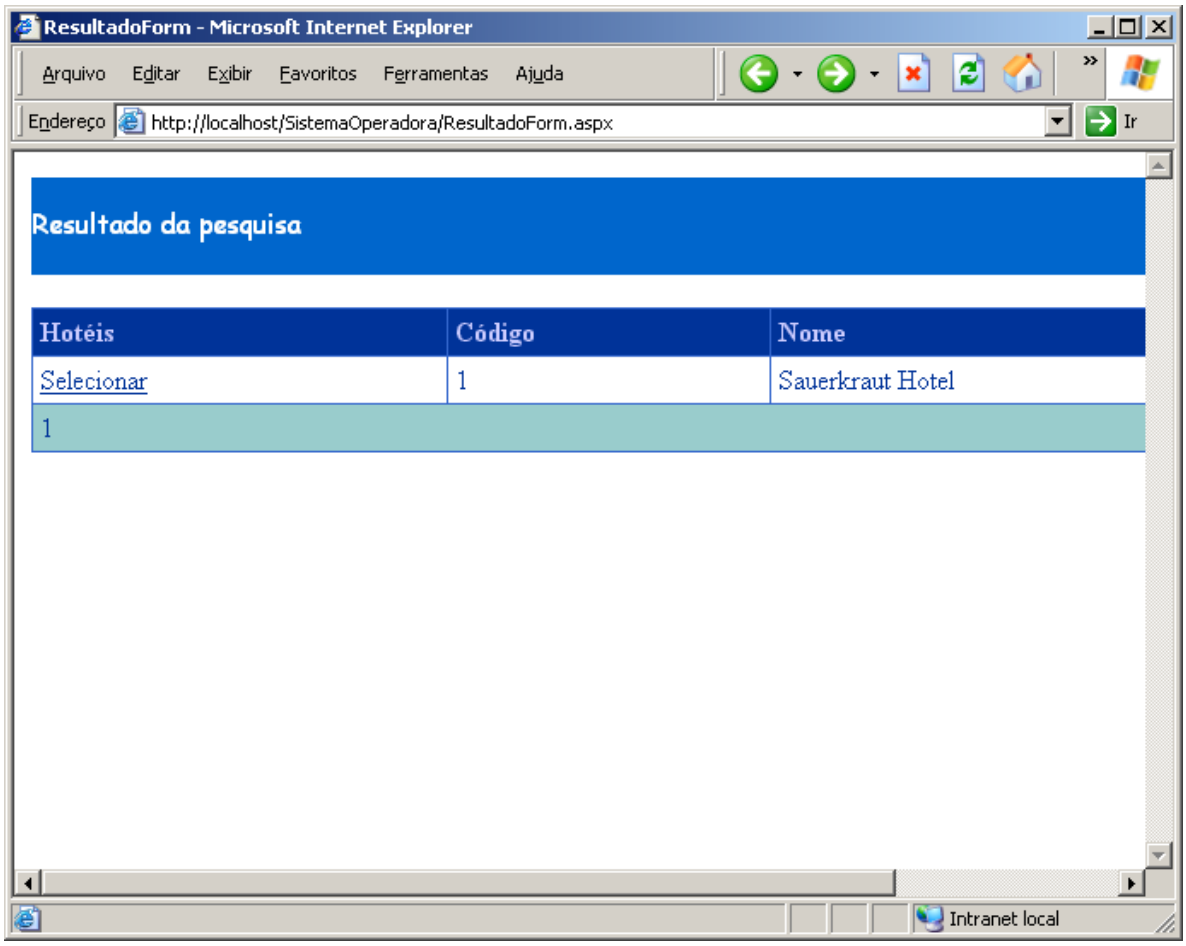


Figura 28: Resultado da pesquisa.

Selecionando um hotel na lista, a página será redirecionada para uma outra página *Web*, esta contendo uma lista com os elementos de locação disponíveis que também atendem os requisitos do filtro informado na tela anterior.

A figura 29 demonstra a lista de elementos de locação retornados no filtro para o hotel selecionado. Na grade ilustrada, pode se perceber os detalhes dos elementos de locação.

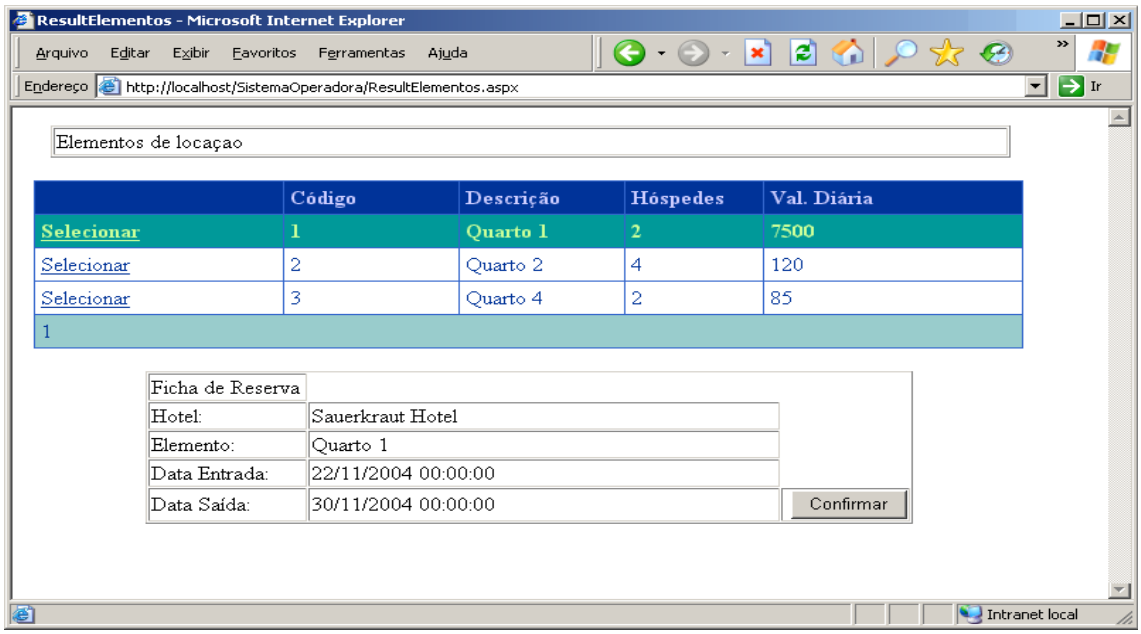


Figura 29: Ficha de Reserva.

Ao selecionar o elemento desejado para reserva, são preenchidos campos para confirmação de reserva. Para confirmar a reserva basta apertar o botão de “Reservar” no canto inferior da tela. Neste momento será enviada uma requisição para o programa principal para o armazenamento de uma reserva.

Para o sistema hotel ter o controle sobre elementos locados e reservados, é necessário atualizar seu banco de dados. As operações efetuadas no componente principal podem ser acessadas pelo devido hotel através da pesquisa de reserva, descrita no caso de uso ”Buscar Informações Reservas”. A tela que implementa esta funcionalidade é representada na figura 30.

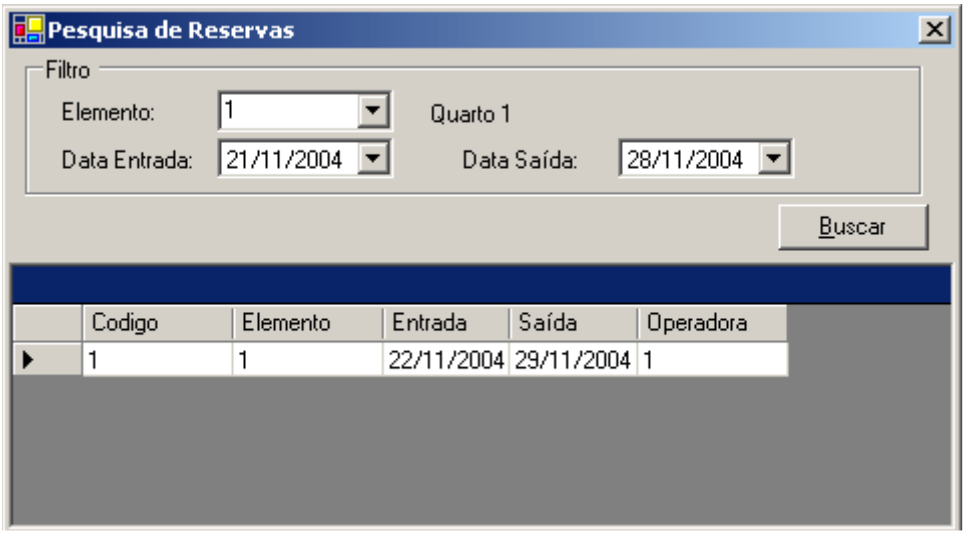


Figura 30: Busca informações de reservas.

5 CONCLUSÕES

O objetivo deste trabalho de criar uma solução para troca de informações entre aplicativos de comércio eletrônico utilizando a tecnologia da plataforma *.NET* foi alcançado. A ferramenta de desenvolvimento *Visual Studio .NET* mostrou-se adequada para a construção de aplicativos *Web Services* utilizando o protocolo SOAP.

A solução desenvolvida permite que aplicações clientes possam efetuar a troca de dados, não importando em qual plataforma estes foram desenvolvidos, desde que estas implementem o SOAP como protocolo de comunicação. Isto pôde ser demonstrado nas aplicações desenvolvidas neste trabalho como o módulo interface hotel e módulo interface operadora, que comprovam a interoperabilidade da solução implementada, e também pode ser demonstrado o funcionamento das aplicações em plataformas de execução diferente entre os clientes.

O *Web Service* pode se entender como uma biblioteca de ligação dinâmica (DLL), porém, com a diferença de que ele pode estar localizado em qualquer lugar do planeta, e os aplicativos podem acessá-lo como se estivessem instalados na própria máquina. Isso torna o desenvolvimento de programas clientes que utilizem este recurso mais fácil e rápido.

O protocolo SOAP possui um grande benefício, ele é aberto e foi adotado pela grande maioria das grandes empresas de *hardware* e *software*. Sua especificação foi submetida ao W3C. O mesmo mostrou-se adequado para implementação de *Web Services* utilizando documentos XML. O XML é ideal para comunicação entre redes heterogêneas, porque suas propriedades baseadas em texto tornam-no independente de plataformas.

Os diagramas UML, desenvolvidos com a ferramenta *Visio*, foram de suma importância para um melhor entendimento e melhor especificação do aplicativo desenvolvido. Os diagramas permitiram ter uma visão mais clara do projeto assim como identificar possíveis falhas mesmo antes do desenvolvimento da aplicação.

A linguagem de desenvolvimento C# mostrou-se muito eficiente e poderosa para o desenvolvimento das aplicações, tanto aplicações clientes de *Windows Forms* e *Web Forms* quanto para o desenvolvimento do *Web Service* utilizando-se do banco de dados.

Este trabalho teve o intuito de demonstrar o resultado da integração de aplicações, não atendo-se ao nível de atualização de banco de dados considerando bases distribuídas entre as entidades por parte dos clientes.

5.1 EXTENSÕES

Como extensão deste trabalho pode-se buscar desenvolver os aplicativos clientes utilizando Banco de Dados afim de que funcione como uma aplicação para uso real.

Pode-se ainda implementar aplicações clientes que consumam o *Web Service* a partir de outras linguagens que também suportem o protocolo SOAP.

REFERÊNCIAS BIBLIOGRÁFICAS

BRAGAGNOLO, Daniel H. **Web Services no mercado**. (2004) Disponível em: <http://www.msdn.com.br/secure/sharepedia/download.aspx?id=52010> Acesso em: 25 mar. 2004.

BRAGAGNOLO, Daniel H. **Visão geral sobre a Plataforma .*NET***. (2003) Disponível em: <http://www.msdn.com.br/secure/sharepedia/download.aspx?id=2750> Acesso em: 13 ago. 2004.

COLPANI, Cristiano Fornari. **Protótipo de *Software* para troca de dados entre aplicações de comercio eletrônico utilizando o protocolo SOAP**. 2002. Monografia (Conclusão de Curso de Bacharel em Ciências da Computação).Universidade Regional de Blumenau. Blumenau, 2002.

FRANCO, Carlos F. **e-Business, Tecnologia de informação e Negócios na Internet.** São Paulo: Atlas, 2001.

FURLAN, José Davi. **Modelagem de Objetos através da UML – *the Unified Modeling Language***. São Paulo: Makron *Books*, 1998.

HOLZNER, Steven. **Desvendando XML.** Rio de Janeiro: Campus, 2001.

LEOPOLDO, Marcus R. M..**Entendendo *o Simple Object Access Protocol* (SOAP)**. (2003) Disponível em <http://www.msdn.com.br/secure/sharepedia/arquivos/SOAP.pdf> . Acesso em: 15 ago.2004.

MICROSOFT. **Developing Microsoft *ASP .NET Web Application Using Visual Studio .NET*.** Estados Unidos: Microsoft Corporation, 2002.

MICROSOFT. **Casos de sucesso, *.NET* como plataforma de negócios** (2004). Disponível em: http://www.microsoft.com/brasil/Casos/interna.aspx?id=176. Acesso em: 9 set. 2004.

RAY, Erick T. **Aprendendo XML**;. Rio de Janeiro: Campus, 2001.

SEELY, Scott. **SOAP: Cross platform web service development using XML**. Upper Saddle River: Pratice Hall PTR, 2002.

SCRIBNER, Kenn; STIVER, Mark C. **Applied SOAP**: implementing .*NET* XML *web services*. Indianapolis: Sams, 2001.

SILLS, Adam *et al*. **XML .NET**: developer´s guide. Rockland: Syngress, 2002.

SKONNARD, Aaron; GUDGIN, Martin. **Essential XML quick reference:** a programmer's XML reference, XPath, XSLT, XML Schema, SOAP and more. Indianápolis: Perason Education, 2002.

SNELL, James; TIDWELL, Doug; KULCHENKO, Pavel.**ProgrammingWeb Services with SOAP.** Sebastopol: O´Reilly, 2001.

SOUSA, Gleiber Martins. **Introdução à linguagem XML**. (2004). Disponível em <http://www.msdn.com.br/secure/sharepedia/download.aspx?id=54573>. Acesso em: 26 jul. 2004.

TOLOMELLI, Leonardo. **Web Services: um novo modelo de negócios para Internet**. (2004). Disponível em <http://www.msdn.com.br/colunas/batepapo/col_batepapo_3.aspx >. Acesso em: 25 mar. 2004.