

**CENTRO UNIVERSITÁRIO FEEVALE**

**ROBERTO KRUG**

**Desenvolvimento de Sistemas com a Tecnologia Multicamadas:  
um Estudo de Caso**

**Novo Hamburgo, dezembro de 2004.**

**ROBERTO KRUG**

**Desenvolvimento de Sistemas com a Tecnologia Multicamadas:  
um Estudo de Caso**

**Centro Universitário Feevale  
Instituto de Ciências Exatas e Tecnológicas  
Curso de Ciência da Computação  
Trabalho de Conclusão de Curso**

**Professor orientador: Ms. Edvar Bergmann Araujo**

**Novo Hamburgo, dezembro de 2004.**

DADOS INTERNACIONAIS DE CATALOGAÇÃO NA PUBLICAÇÃO (CIP)

Centro Universitário Feevale, RS, Brasil

Krug, Roberto

Desenvolvimento de sistemas com a tecnologia multicamadas: um estudo de caso / Roberto Krug; Orientador Edvar Bergmann Araujo - Novo Hamburgo : Feevale, 2004.

114f. : il.; 28cm.

Inclui bibliografia.

Trabalho de conclusão do curso de Ciência da Computação do Centro Universitário Feevale.

1. Multicamadas - Arquitetura de computador 2. Negócios - Regras  
3. Sistemas de informação gerencial - I. Araújo, Edvar Bergmann II.  
Título.

CDU 004.451:658

Bibliotecária responsável: Gina Maria da Gama CRB 10/1478

## **Dedicatória**

Dedico este trabalho à querida e amada esposa Ani Moni Dietrich Krug e a linda filha Rebeca Dietrich Krug, por darem um significado especial a minha vida. Vocês foram o meu maior apoio e incentivo na conquista deste nobre objetivo.

“A estrada para o sucesso não é uma reta. Há uma curva chamada fracasso, um trevo chamado confusão, um quebra-molas chamado amigos, faróis de advertência chamados família e pneus furados chamados empregos. Mas... se você tiver um estepe chamado determinação, um motor chamado perseverança, um seguro chamado fé e um motorista chamado Jesus, você chegará a um lugar chamado sucesso!”

Autor desconhecido

## **Agradecimentos**

Em primeiro lugar quero agradecer a Deus, que esteve comigo em todos os momentos, tanto nas horas frias como nas mais calorosas. E por continuar ao meu lado, dando-me orientação nesta nova jornada.

O meu agradecimento a todos aqueles que me ajudaram a chegar até aqui. A minha mãe, ao meu pai (*in memoriam*) e principalmente a minha esposa, por ter acreditado, me apoiado e que deu-me direção, em mais uma conquista da vida. Também o meu agradecimento a minha filha que veio fazer parte da nossa família, com seu sorriso e ternura.

Um agradecimento especial ao meu professor Ms. Edvar Bergmann Araujo e a todos os outros professores, pela dedicação e comprometimento dispensados, e não só pelos conhecimentos a mim transmitidos, mas também pelo apoio nos momentos difíceis.

Também quero agradecer a todos os meus colegas, pela caminhada e pela conquista de mais um objetivo, que juntos alcançamos.

## Resumo

A automatização do ambiente empresarial, a integração de tarefas, o aumento do número de usuários, a atualização constante dos Sistemas e em diferentes pontos da rede, têm afetado de forma prejudicial o desempenho, a segurança, a escalabilidade e a confiabilidade dos Sistemas. Neste contexto, a Arquitetura de Sistemas Multicamadas surge como uma proposta para minimizar estes problemas.

A arquitetura multicamadas é um modelo de programação que prevê a divisão do programa em três ou mais partes bem definidas e distintas, sendo que as três partes principais são: Interface (apresentação), Regras de Negócio (lógica) e Persistência (banco de dados). Nos sistemas tradicionais (Cliente/Servidor), também conhecidos como sistema de duas camadas, há três opções com relação às Regras de Negócio: colocar junto da interface do usuário, junto ao banco de dados ou mesclar as duas opções. No entanto, nenhuma dessas opções é considerada como uma solução satisfatória.

Com a introdução da camada de lógica, as Regras de Negócio concentram-se no Servidor de Aplicações. Como são nestas regras que ocorre a maior partes das mudanças e uma vez que elas estão centralizadas no Servidor de Aplicações, resolve-se o problema da atualização, em centenas ou milhares de computadores, cada vez que uma Regra de Negócio for alterada, facilitando em muito a atualização dos sistemas.

Sendo assim, este trabalho tem como objetivo analisar o funcionamento da arquitetura de Sistemas Multicamadas e aplicar este modelo através do projeto e implementação de módulos de um Sistema ERP (*Enterprise Resource Planning*) usando esta tecnologia.

**Palavras-chave:** Multicamadas, Regras de Negócio, ERP.

## **Abstract**

Title: Development of Systems with the Multitier Technology - a Study of Case

The automation of the enterprise environment, the integration of tasks, the increase of the number of users, the constant update of the systems and in different points of the network, have harmfully affected the performance, safety, scaling, and the reliability of the Systems. In this context, the Architecture of the Multitier Systems shows up as a proposal of minimizing these problems.

The multitier architecture is a programming model that foresees the division of the program in three or more well defined and distinct parts, of which the three main parts are: Interface (presentation), Business rules (logic) and Persistence (database). In the traditional systems (Client/Server), also known as two tier system, there are three options concerning business rules: put on the user's interface, on the database or mix up both options. Though none of these options is considered as a satisfactory solution.

With the insertion of the logic tier, the business rules focus on the Application Server. As most changes happen in these rules and since they are focused on the Application Server, one solves, for instance, the update problem, everytime a Business rule is changed, making the systems update a lot easier.

So this work aims to analyse the functioning of the Multitier System architecture and apply this model through the project and insertion of modules of an ERP System (Enterprise Resource Planning) using this technology.

**Keywords:** Multitier, Business Rules, ERP.

## Lista de Figuras

Figura 2.1 – Modelo de Sistema Centralizado _____	22
Figura 2.2 – Estrutura física do modelo em Duas Camadas _____	23
Figura 2.3 – Estrutura física do modelo em Multicamadas _____	27
Figura 2.4 – Estrutura lógica do modelo Multicamadas _____	28
Figura 2.5 – Estrutura física do modelo em Quatro Camadas _____	29
Figura 2.6 – Tecnologia Mista _____	30
Figura 2.7 – Negócio e Dados no mesmo Servidor _____	33
Figura 2.8 – Negócio e Dados em Servidores diferentes _____	34
Figura 2.9 – Negócio e Dados em dois Servidores idênticos _____	35
Figura 2.10 – Negócio e Dados utilizando um Banco de Dados particionado _____	36
Figura 3.1 – A arquitetura CORBA _____	39
Figura 3.2 – Comunicação de Objetos Distribuídos _____	43
Figura 4.1 – Os Componentes DataSetProvider e ClientDataSet _____	46
Figura 4.2 – Componentes DataSnap do Delphi _____	47
Figura 4.3 – Arquitetura DataSnap para aplicações Multicamadas _____	47
Figura 4.4 – Criação de um WebModule com o WebServices _____	49
Figura 4.5 – Exemplo da definição de uma Interface _____	50
Figura 4.6 – Exemplo da implementação de uma Interface _____	51
Figura 4.7 – Criação de um Cliente com o WebServices _____	52
Figura 4.8 – Configuração do componente THTTPIO _____	53
Figura 4.9 – Código do Botão Soma _____	53
Figura 5.1 – Evolução da UML _____	57
Figura 5.2 – A Evolução mais Recente _____	58
Figura 5.3 – Diagrama de Caso de Uso _____	59

Figura 5.4 – Diagrama de Classes _____	60
Figura 5.5 – Diagrama de Seqüência _____	61
Figura 5.6 – Visões de um Sistema _____	63
Figura 6.1 – Metodologia Espiral _____	65
Figura 6.2 – ERP – Informação e Conhecimento _____	67
Figura 6.3 – Classe de Usuários _____	68
Figura 6.4 – Classe de Clientes _____	68
Figura 6.5 – Classe de Fornecedores _____	69
Figura 6.6 – Classe de Representantes _____	69
Figura 6.7 – Classe de Transportadoras _____	70
Figura 6.8 – Classe das Naturezas de Operação _____	70
Figura 6.9 – Classe dos Produtos _____	71
Figura 6.10 – Classe das Formas de Pagamento _____	71
Figura 6.11 – Classe das Cidades _____	72
Figura 6.12 – Classe dos Estados _____	72
Figura 6.13 – Classe dos Países _____	72
Figura 6.14 – Classe dos Bancos _____	73
Figura 7.1 – Caso de Uso - Controle de Pedido _____	76
Figura 7.2 – Diagrama de Classes - Controle de Pedido _____	77
Figura 7.3 – Diagrama de Seqüência - Controle de Pedido _____	78
Figura 7.4 – Caso de Uso - Emissão da Nota Fiscal _____	78
Figura 7.5 – Diagrama de Classes - Emissão da Nota Fiscal _____	80
Figura 7.6 – Diagrama de Seqüência - Emissão da Nota Fiscal _____	81
Figura 8.1 – Caso de Uso - Contas a Receber _____	84
Figura 8.2 – Diagrama de Classes - Contas a Receber _____	84

Figura 8.3 – Diagrama de Seqüência - Contas a Receber _____	85
Figura 8.4 – Caso de Uso - Contas a Pagar _____	86
Figura 8.5 – Diagrama de Classes - Contas a Pagar _____	86
Figura 8.6 – Diagrama de Seqüência - Contas a Pagar _____	87
Figura 9.1 – Estrutura do Sistema ERP _____	89
Figura 9.2 – Tela inicial para criação do Servidor de Aplicação _____	90
Figura 9.3 – Servidor de Aplicação tipo CGI (Executável) _____	90
Figura 9.4 – Web Module do ERP _____	91
Figura 9.5 – Web Services - Data Module ERP _____	92
Figura 9.6 – Cadastro de Clientes - Modo de Implementação _____	92
Figura 9.7 – Aplicativo IIS (Internet Information Services) _____	93
Figura 9.8 – Diagrama E-R do Faturamento / Financeiro _____	94
Figura 9.9 – Web Services - Data Module ERP _____	101
Figura 9.10 – Regra de Negócio - Valida CPF _____	102
Figura 9.11 – Regra de Negócio - Gera ID _____	102
Figura 9.12 – ERP - Tela Inicial _____	103
Figura 9.13 – Tela Principal do Cadastro de Clientes _____	104
Figura 9.14 – Tela de Manutenção do Cadastro de Clientes _____	104
Figura 9.15 – Tela Principal dos Pedidos _____	105
Figura 9.16 – Tela de Manutenção dos Pedidos _____	106
Figura 9.17 – Tela Principal da Emissão de Nota Fiscal _____	107
Figura 9.18 – Tela de Manutenção da Emissão de Nota Fiscal _____	108
Figura 9.19 – Tela Principal do Contas a Receber _____	109
Figura 9.20 – Tela de Manutenção do Contas a Receber _____	109
Figura 9.21 – Web Services - Data Module ERP _____	110

## Lista de Tabelas

Tabela 9.1 – Estrutura da entidade Pedidos _____	95
Tabela 9.2 – Estrutura da entidade Pedidos_Itens _____	95
Tabela 9.3 – Estrutura da entidade Notas _____	96
Tabela 9.4 – Estrutura da entidade Notas_Itens _____	97
Tabela 9.5 – Estrutura da entidade Duplicatas _____	97
Tabela 9.6 – Estrutura da entidade Clientes _____	98
Tabela 9.7 – Estrutura da entidade Fornecedores _____	98
Tabela 9.8 – Estrutura da entidade Representantes _____	98
Tabela 9.9 – Estrutura da entidade Transportadoras _____	99
Tabela 9.10 – Estrutura da entidade Produtos _____	99
Tabela 9.11 – Estrutura da entidade Naturezas _____	99
Tabela 9.12 – Estrutura da entidade Formas_Pagamento _____	99
Tabela 9.13 – Estrutura da entidade Bancos _____	100
Tabela 9.14 – Estrutura da entidade Usuarios _____	100
Tabela 9.15 – Estrutura da entidade Cidades _____	100
Tabela 9.16 – Estrutura da entidade Estados _____	100
Tabela 9.17 – Estrutura da entidade Países _____	100

## Lista de Abreviaturas

<b>ASP</b>	<b>Active Server Pages</b>
<b>CGI</b>	<b>Common Gateway Interface</b>
<b>CNAB</b>	<b>Centro Nacional de Automação Bancária</b>
<b>CNPJ</b>	<b>Cadastro Nacional de Pessoa Jurídica</b>
<b>COM</b>	<b>Component Object Model</b>
<b>COM+</b>	<b>Component Object Model Plus</b>
<b>CORBA</b>	<b>Common Object Request Broker Architecture</b>
<b>DCOM</b>	<b>Distributed Component Object Model</b>
<b>DLL</b>	<b>Dynamic Link Libraries</b>
<b>EDI</b>	<b>Eletronic Data Interchange</b>
<b>ERP</b>	<b>Enterprise Resource Planning</b>
<b>EXE</b>	<b>EXEcutável</b>
<b>HTTP</b>	<b>Hiper Text Transfer Protocol</b>
<b>I/O</b>	<b>Input/Ouput</b>
<b>IDL</b>	<b>Interface Definition Language</b>
<b>ICMS</b>	<b>Imposto sobre Circulação de Mercadorias e Serviços</b>
<b>IIS</b>	<b>Internet Information Services</b>
<b>IIOP</b>	<b>Internet Inter-ORB Protocol</b>
<b>IPI</b>	<b>Imposto sobre Produtos Industrializados</b>
<b>J2EE</b>	<b>Java 2 Entreprise Edition</b>
<b>NetBEUI</b>	<b>NetBios Enhanced User Interface</b>
<b>MIDAS</b>	<b>Middle-tier Distributed Applications Services</b>
<b>MSDTC</b>	<b>Microsoft Distributed Transaction Coordinator</b>
<b>MTS</b>	<b>Microsoft Transation Server</b>
<b>OLE</b>	<b>Object Linking and Embedding</b>
<b>OMG</b>	<b>Object Management Group</b>
<b>OMT</b>	<b>Object Modeling Technique</b>
<b>OO</b>	<b>Object Oriented</b>
<b>OOSE</b>	<b>Object Oriented Software Engineering</b>
<b>ORB</b>	<b>Object Request Broker</b>
<b>OS/2</b>	<b>Operating System/2</b>
<b>RAD</b>	<b>Rapid Application Development</b>
<b>TCP/IP</b>	<b>Transmission Control Protocol/Internet Protocol</b>
<b>RMI</b>	<b>Remote Method Invocation</b>
<b>RPC</b>	<b>Remote Procedure Call</b>
<b>SOAP</b>	<b>Simple Object Access Protocol</b>
<b>SPX</b>	<b>Sequenced Packet eXchange</b>
<b>UML</b>	<b>Unified Modeling Language</b>
<b>W3C</b>	<b>Word Wide Web Consortium</b>
<b>WSDL</b>	<b>Web Service Description Language</b>
<b>XML</b>	<b>eXtensible Markup Language</b>

# Sumário

<b>1 INTRODUÇÃO</b>	<b>15</b>
<b>1.1 TEMA</b>	<b>15</b>
<b>1.2 MOTIVAÇÃO</b>	<b>15</b>
<b>1.3 OBJETIVOS</b>	<b>19</b>
<b>1.4 ESTRUTURA DO TEXTO</b>	<b>19</b>
<b>2 ARQUITETURA DE SISTEMAS EM CAMADAS</b>	<b>21</b>
<b>2.1 UMA CAMADA</b>	<b>22</b>
<b>2.2 DUAS CAMADAS</b>	<b>23</b>
<b>2.3 MULTICAMADAS</b>	<b>25</b>
2.3.1 INTERFACE	26
2.3.2 REGRAS DE NEGÓCIO	26
2.3.3 PERSISTÊNCIA	27
<b>2.4 A QUARTA CAMADA</b>	<b>28</b>
<b>2.5 TECNOLOGIA MISTA</b>	<b>29</b>
<b>2.6 CLIENTE/SERVIDOR X MULTICAMADAS</b>	<b>31</b>
2.6.1 DESVANTAGENS DO CLIENTE/SERVIDOR	31
2.6.2 VANTAGENS DAS MULTICAMADAS	31
<b>2.7 MODELOS DE DISTRIBUIÇÃO DA APLICAÇÃO</b>	<b>33</b>
<b>3 PROTOCOLOS DE COMUNICAÇÃO</b>	<b>37</b>
<b>3.1 CORBA</b>	<b>37</b>
3.1.1 CARACTERÍSTICAS	38
3.1.2 ARQUITETURA	38
3.1.3 INTERFACES	39
<b>3.2 COM</b>	<b>39</b>
3.2.1 DCOM	40
3.2.2 MTS	40
3.2.3 COM+	41
<b>3.3 SOAP</b>	<b>42</b>
<b>3.4 JAVA RMI</b>	<b>43</b>
3.4.1 COMUNICAÇÃO DE OBJETOS DISTRIBUÍDOS	43
<b>3.5 CONSIDERAÇÕES FINAIS</b>	<b>44</b>
<b>4 MULTICAMADAS NO DELPHI</b>	<b>45</b>
<b>4.1 DATASETPROVIDER</b>	<b>45</b>
<b>4.2 DATASNAP</b>	<b>46</b>
<b>4.3 WEB SERVICES</b>	<b>47</b>
4.3.1 IMPLEMENTANDO UM SERVIDOR COM WEB SERVICES EM DELPHI	48
4.3.2 IMPLEMENTANDO CLIENTES COM WEB SERVICES	52
<b>4.4 CONSIDERAÇÕES FINAIS</b>	<b>54</b>

<b>5 UML</b>	<b>55</b>
<b>5.1 EVOLUÇÃO DA UML</b>	<b>56</b>
<b>5.2 DIAGRAMAS DA UML</b>	<b>58</b>
5.2.1 ANÁLISE	59
5.2.2 PROJETO	60
5.2.3 IMPLEMENTAÇÃO	62
<b>5.3 AS VISÕES DE UM SISTEMA</b>	<b>62</b>
<b>6 ESTUDO DE CASO</b>	<b>64</b>
<b>6.1 METODOLOGIA DE DESENVOLVIMENTO</b>	<b>65</b>
<b>6.2 REQUISITOS DO SOFTWARE</b>	<b>66</b>
<b>6.3 MÓDULOS DO SISTEMA ERP</b>	<b>66</b>
6.3.1 CADASTROS BÁSICOS	67
6.3.2 FATURAMENTO	73
6.3.3 FINANCEIRO	73
<b>7 FATURAMENTO</b>	<b>74</b>
<b>7.1 CONTROLE DE PEDIDO</b>	<b>75</b>
<b>7.2 EMISSÃO DA NOTA FISCAL</b>	<b>78</b>
<b>8 FINANCEIRO</b>	<b>82</b>
<b>8.1 CONTAS A RECEBER</b>	<b>83</b>
<b>8.2 CONTAS A PAGAR</b>	<b>85</b>
<b>9 IMPLEMENTAÇÃO</b>	<b>88</b>
<b>9.1 TECNOLOGIA / ESTRUTURA</b>	<b>89</b>
<b>9.2 MAPEAMENTO OBJETO-RELACIONAL</b>	<b>93</b>
<b>9.3 IMPLEMENTAÇÃO DO SOFTWARE</b>	<b>101</b>
9.3.1 REGRAS DE NEGÓCIO	101
9.3.2 APRESENTAÇÃO	103
9.3.3 PERSISTÊNCIA	110
<b>CONCLUSÃO</b>	<b>111</b>
<b>REFERÊNCIAS BIBLIOGRÁFICAS</b>	<b>113</b>

# **1 INTRODUÇÃO**

## **1.1 Tema**

O tema principal deste trabalho é analisar o funcionamento da arquitetura de Sistemas Multicamadas e aplicar este modelo através do projeto e implementação de um software.

## **1.2 Motivação**

O ambiente empresarial atual é extremamente competitivo. Há vários anos os gestores vêm procurando formas de melhorar processos, reduzir custos e qualificar produtos e serviços. Os Sistemas de Informação têm papel fundamental neste contexto, uma vez que estes trazem contribuições significativas para a melhoria e o controle de processos, qualificando os produtos e serviços oferecidos pela organização e, em decorrência disto, tornam a empresa mais competitiva.

A automatização do ambiente empresarial, a integração de tarefas, o aumento do número de usuários, a atualização constante dos Sistemas e em diferentes pontos da rede, têm afetado de forma prejudicial o desempenho, a segurança, a escalabilidade e a confiabilidade dos Sistemas (RODRIGUES, 2002. p.10-14).

Os desenvolvedores e gerentes de projeto têm buscado usar novas técnicas de programação para que os sistemas atendam as premissas de velocidade, segurança, escalabilidade e confiabilidade. Uma destas técnicas é o modelo da aplicação em multicamadas e orientada a transações. Elas os auxiliam a lidar com estruturas de dados complexas, alterações em regras de negócio e mudanças em relação à necessidade dos usuários (CRAWFORD, 2004).

A arquitetura Multicamadas pode ser considerada como uma evolução de outras arquiteturas. Num primeiro momento, o qual foi considerado o período dos sistemas de somente uma camada, os softwares executavam em computadores conhecidos como *Mainframes*. As estações ligadas ao computador principal eram chamadas de terminais “burros”, pois estes terminais, não tinham memória, nem processador e nem disco rígido e, por isto, não realizavam nenhum tipo de processamento. Como consequência disto, o servidor era sobrecarregado com todo o processamento (SALEMI, 1995).

Em um segundo momento, surgiu o modelo de sistemas Cliente/Servidor, também conhecido como o modelo de duas camadas, por transferir parte do processamento que era feito no servidor para as estações, pois estas já tinham memória, processador e disco rígido (WILDEROM, 2002).

Como uma evolução do modelo de duas camadas, surgiu, com o crescimento constante da Internet, o modelo de três camadas, também conhecido como multicamadas, ncamadas ou ainda como *multitier*. A idéia básica do modelo de três camadas é: "retirar" as Regras de Negócio (lógica da aplicação) do cliente (estação) e centralizá-las em um determinado ponto, o qual é chamado de Servidor de Aplicações. O acesso ao banco de dados é feito através das regras contidas no Servidor de Aplicações. Ao centralizar as Regras de Negócio em um único ponto, torna-se mais fácil a atualização destas regras (RODRIGUES, 2002. p.7-9).

Com a introdução da camada de Lógica resolve-se o problema de ter que atualizar a aplicação, em centenas ou milhares de computadores, cada vez que uma Regra de Negócio for alterada.

De acordo com Battisti (2001)<sup>1</sup> e Rodrigues (2002), um detalhamento maior do novo modelo de aplicação e as três camadas são as seguintes:

- **Interface:** a camada da Interface, também conhecida como “Apresentação”, permanece no cliente, ou melhor, no aplicativo da estação (Cliente/Servidor).
- **Regras de Negócio:** as Regras de Negócio, também conhecidas como “Lógica”, são as quais determinam de que maneira os dados serão utilizados. Esta camada foi deslocada para o Servidor de Aplicações. Desta maneira, quando uma regra de

---

<sup>1</sup> Disponível em < <http://www.juliobattisti.com.br/artigos/ti/ncamadas.asp>>. Acesso em 24 mar. 2004.

negócio for alterada, basta atualizá-la no Servidor de Aplicações. Após a atualização todos os usuários passarão a ter acesso a nova versão, sem que seja necessário reinstalar o programa em cada um dos computadores da rede, facilitando a tarefa de manter o sistema atualizado.

- **Persistência:** a camada de Persistência fica no servidor do banco de dados, na qual reside toda a informação necessária para o funcionamento da aplicação. Cabe ressaltar, que os dados somente são acessados através do Servidor de Aplicação, e não diretamente pela aplicação do cliente (estação).

Ao longo dos anos, ferramentas têm sido criadas para agilizar, facilitar e melhorar a montagem dos projetos e o desenvolvimento das aplicações. Com estas ferramentas, tornou-se indispensável a utilização de componentes, bibliotecas e padrões de projeto visando a produtividade no processo de desenvolvimento dos *softwares* e a aceitação dos mesmos, ainda mais neste mercado que é altamente competitivo. Prova disto é que grandes fornecedores de ferramentas de desenvolvimento, tais como: Sun, Borland, Microsoft, Oracle, entre outros, têm lançado freqüentes versões, incluindo cada vez mais recursos.

A Borland possui uma série de ferramentas de desenvolvimento de *software*. Dentre elas, o Delphi é uma das mais utilizadas no Brasil. Conforme Rodrigues, o modelo de multicamadas foi introduzido no Delphi 3, melhorado no Delphi 5 e já no Delphi 7, foram lançados diversos recursos adicionais, novos componentes e facilidades, tais como o *DataSnap* com suporte a SOAP-HTTP/XML<sup>2</sup> (*Simple Object Access Protocol - Hyper Text Transfer Protocol / Extensible Markup Language*) (RODRIGUES, 2002. p.7-9).

“DataSnap é uma coletânea de tecnologias que funcionam em conjunto e tem o objetivo de facilitar o desenvolvimento de aplicativos distribuídos” (RODRIGUES, 2002. p.8). Com o *DataSnap*, os dados ficam armazenados em um banco de dados, no Servidor de Banco de Dados, mas estes nunca são acessados diretamente pelo cliente. Ao invés disso, o cliente utiliza objetos COM/COM+ (*Component Object Model / Component Object Model Plus*) disponíveis nos servidores da rede. Os objetos COM/COM+ (camada de lógica) contêm toda a lógica de acesso aos dados. O cliente faz a solicitação para o objeto COM/COM+ e este

---

<sup>2</sup> HTTP – *Hyper Text Transfer Protocol* (Protocolo de Transferência de Hipertexto) é o protocolo básico para o trânsito de dados e informações na Internet.  
XML – *eXtensible Markup Language* é o formato universal dos documentos e dados disponíveis na Internet.

acessa os dados de acordo com a lógica definida no próprio objeto. O objeto COM/COM+ recebe a resposta do servidor de banco de dados e repassa a resposta ao cliente.

“O cliente pode ser instalado nas estações do usuário ou, simplesmente, ser uma página Web, armazenada no servidor IIS - *Internet Information Services* (camada de aplicação)” (BATTISTI, 2001)<sup>3</sup>.

A tecnologia J2EE (*Java 2 Enterprise Edition*) da Sun, também oferece um modelo de aplicação distribuída multicamada, com a capacidade para reutilização de componentes, troca de dados integrada baseada em XML e conta com um modelo de segurança unificado e um flexível controle de transações. Com a tecnologia J2EE é possível desenvolver soluções baseadas em componentes, independente de plataformas e que não estarão vinculadas a um banco de dados específico.

Baseado neste novo modelo de sistemas, multicamadas, este trabalho se propõe a: (i) identificar características, técnicas e ferramentas para o desenvolvimento de *software* nesta arquitetura e (ii) aplicar estes conceitos através da análise, projeto e implementação de módulos de um sistema ERP nesta arquitetura. Será enfatizada à camada das Regras de Negócio, pois é nesta camada que se concentra a principal diferença em relação a um sistema de duas camadas (Cliente/Servidor).

Os módulos desenvolvidos serão o “Faturamento” e o “Financeiro”. A escolha destes módulos se dá pela importância dos mesmos na gestão dos negócios das empresas. Com o desenvolvimento desses módulos, as empresas poderão via sistema emitir, entre outros documentos, as notas fiscais, as duplicatas e os bloquitos. Também terão um controle mais exato e eficiente de suas contas financeiras (a pagar, e a receber).

---

Disponível em <<http://www.dicweb.com/>>. Acesso em 06 jun. 2004.

<sup>3</sup> Disponível em <[http://www.timaster.com.br/revista/colunistas/ler\\_colunas\\_emp.asp?cod=507](http://www.timaster.com.br/revista/colunistas/ler_colunas_emp.asp?cod=507)>. Acesso em 24 mar. 2004.

### **1.3 Objetivos**

O objetivo principal deste trabalho é analisar o funcionamento da arquitetura de sistemas multicamadas e aplicar este modelo através do projeto e implementação de um software usando esta tecnologia.

Como objetivos específicos destacam-se:

1. Identificar os princípios e as características da arquitetura multicamadas.
2. Analisar a tecnologia, as ferramentas e os componentes existentes para desenvolver um sistema multicamadas.
3. Levantar os requisitos para o desenvolvimento do software.
4. Projetar o software.
5. Desenvolver o software.

### **1.4 Estrutura do Texto**

O capítulo dois apresenta as características da arquitetura Multicamadas e as características das arquiteturas anteriores. São enfatizadas as divisões por camada e a utilização de cada uma delas. Além disto, este capítulo também apresenta as vantagens e desvantagens do Cliente/Servidor e Multicamadas e destaca diferentes formas de como estas camadas podem ser organizadas dentro da estrutura física de uma empresa.

O capítulo três discorre sobre os protocolos de comunicação que são utilizados para fazer a comunicação entre as camadas de um sistema. São apresentados os protocolos CORBA, COM, DCOM, MTS, COM+, SOAP, Java RMI e as características de cada um deles. Por fim, comenta-se a comunicação de objetos distribuídos.

O capítulo quatro destaca os recursos e componentes utilizados no Delphi para o desenvolvimento de sistemas na arquitetura Multicamadas. São apresentados os recursos do DataSetProvider, do DataSnap e do Web Services. Este capítulo também inclui a implementação de um Servidor e de um Cliente com *Web Services*.

O capítulo cinco trata da UML, abordando quais os diagramas que a compõe e destacando quais destes diagramas são utilizados nas três das fases do desenvolvimento de um sistema: análise, projeto e implementação.

O capítulo seis aborda o início do estudo de caso proposto. Neste capítulo serão apresentados a metodologia de desenvolvimento, trata da implementação dos cadastros principais e introduz os módulos de faturamento e do financeiro.

Os capítulos sete e oito destacam os requisitos do sistema, respectivamente dos módulos Faturamento e Financeiro. Estes capítulos apresentam os diagramas de caso de uso, os diagramas de classes e os diagramas de seqüência do sistema.

No capítulo nove tem-se a implementação do sistema. A tecnologia usada, a estrutura do sistema e como iniciar um sistema multicamadas com o Delphi através do protocolo SOAP. Este capítulo também destaca algumas telas utilizadas na implementação do sistema.

Por fim, apresenta-se a conclusão do trabalho.

## 2 ARQUITETURA DE SISTEMAS EM CAMADAS

Até pouco tempo, mesmo com o grande avanço tecnológico, com a facilidade da Internet e com a existência de diversas ferramentas para desenvolvimento de Sistemas, não se tinha a disposição ferramentas que facilitassem o projeto de aplicações corporativas e o acesso a dados distribuídos.

Há pouco tempo surgiu no mercado o conceito da arquitetura de aplicações Multicamadas, a qual facilita em muito o desenvolvimento de aplicações distribuídas, proporcionando um melhor desempenho, uma melhor segurança, escalabilidade e uma maior produtividade na implementação e manutenção destas aplicações (PEREZ, 2004)<sup>4</sup>.

Mas, até chegar ao desenvolvimento de sistemas com a arquitetura multicamadas passou-se por duas outras fases e modelos de desenvolvimento. O modelo de uma camada (*Mainframes* e processamento local) e o modelo de duas camadas (Cliente/Servidor). No momento atual, empresas de médio e grande porte estão investindo e partindo para a nova arquitetura do mercado, a de multicamadas. As próximas seções apresentam uma breve descrição destas arquiteturas, destacando vantagens e desvantagens de cada uma delas.

---

<sup>4</sup> Disponível em <<http://www.imasters.com.br/web/conteudo/secao.php?codartigo=1756>>. Acesso em 10 jun. 2004.

## 2.1 Uma Camada

Os sistemas de Uma Camada surgiram com a arquitetura dos *Mainframes*. Esta arquitetura caracteriza-se por dois componentes principais: (i) o computador central (*Mainframe*) e (ii) os terminais de acesso. Os terminais de acesso eram chamados de terminais “burros”, pois não tinham memória, nem processador e nem disco rígido e, sendo assim, não realizavam nenhum tipo de processamento. Em consequência, o servidor ficava sobrecarregado, pois todo o processamento era realizado por ele (figura 2.1) (SALEMI, 1995).

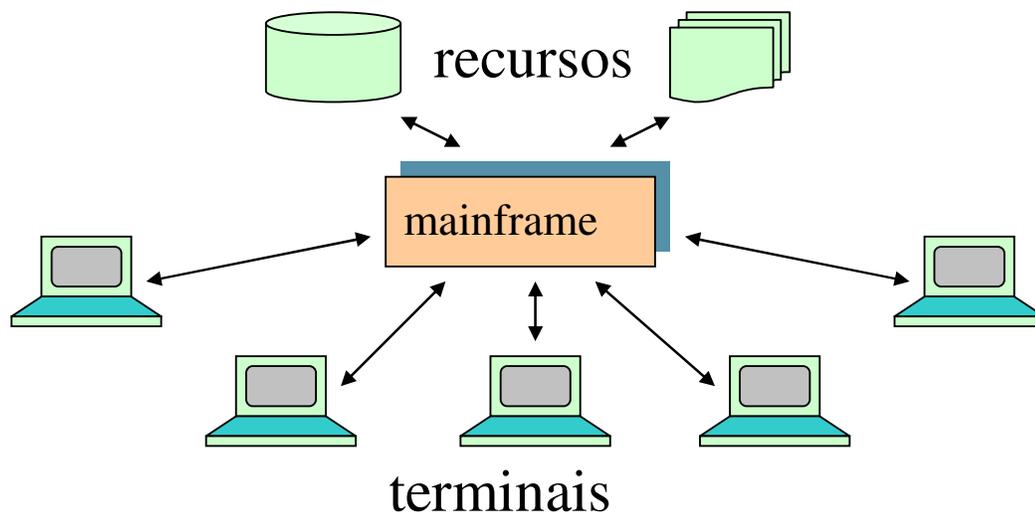


Figura 2.1 – Modelo de Sistema Centralizado

Aplicações que acessam os dados na própria máquina, com a utilização do Access, Paradox e Dbase também são consideradas de somente uma camada (*Single-Tier*), pois o sistema, o banco de dados e o processamento é feito localmente, ou melhor, tudo e qualquer processo é realizado por somente uma máquina (CANTÚ, 1998. p.484 e 876).

Para que os recursos e o processamento pudessem ser compartilhados, e não feitos por somente uma máquina, com possibilidade de sobrecarga e dificultando a escalabilidade, a solução foi partir para a arquitetura cliente/servidor. Nesta arquitetura, parte do processamento que era feito no servidor passa a ser realizado pelas estações (clientes), pois estas passam a ter memória, processador e disco rígido (SALEMI, 1995).

## 2.2 Duas Camadas

A arquitetura de Duas Camadas (*Two-Tier*), também conhecida como Cliente/Servidor, vem substituir a antiga arquitetura: (i) sistema centralizado, na qual os sistemas eram executados nos *Mainframes* com vários terminais interligados e os recursos estavam centralizados e (ii) aplicações locais, nas quais todo o processamento era realizado pelos clientes.

O cliente/servidor é uma arquitetura de rede (sistema distribuído), onde existem dois módulos básicos na rede: o servidor e os clientes. O servidor é uma máquina da rede que é responsável por servir os clientes com o que for solicitado. Clientes são as máquinas que solicitam informações que estão contidas no servidor. Nesta arquitetura, tanto os recursos do servidor como os dos clientes podem ser compartilhados (WILDEROM, 2002).

É no servidor que normalmente ficam os sistemas mais pesados da rede, tais como o banco de dados. As máquinas clientes, são terminais que efetuam processamentos, mas com um poder de processamento menor, pois estas não executam aplicativos que requerem tantos recursos (figura 2.2).

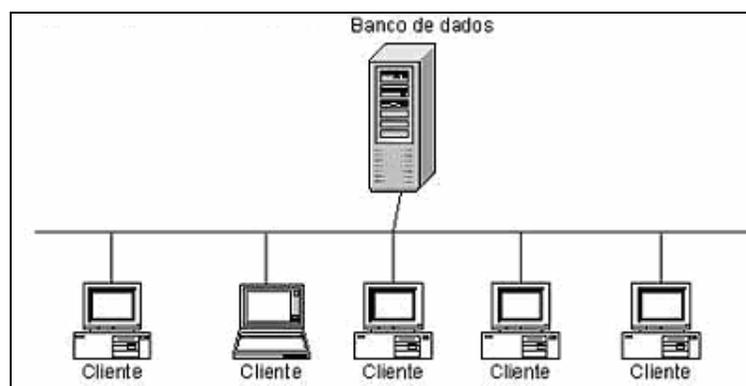


Figura 2.2 – Estrutura física do modelo em Duas Camadas

No cliente/servidor, as aplicações são desenvolvidas utilizando-se um modelo de desenvolvimento em duas camadas. Nesta arquitetura, uma aplicação é instalada em cada estação de trabalho que fará uso do sistema. Esta aplicação faz acesso a um banco de dados que fica residente no Servidor de Banco de dados.

Em relação às Regras de Negócio (camada que faz a principal diferença em Multicamadas), há três opções: colocá-las junto da Interface do Usuário, junto ao Banco de

Dados ou mesclar as duas opções, mas nenhuma dessas opções é considerada como uma solução satisfatória (WILDEROM, 2002).

Em uma arquitetura cliente/servidor as máquinas não precisam ser do mesmo fabricante ou do mesmo tipo, mas para que estas possam ser interligar pela rede, elas precisam usar um protocolo de comunicação, tais como o TCP/IP, NetBEUI, SPX, etc...

Algumas características que podem ser destacadas em relação ao Servidor:

- **Processamento especializado:** como é uma máquina servidora, esta normalmente é uma máquina mais robusta e com maior capacidade de processamento;
- **Atendimento a clientes simultâneos:** as solicitações dos clientes podem ser processadas simultaneamente, realizando assim, o controle de concorrência;
- **Sistema Operacional robusto:** normalmente se usa um sistema operacional com características como segurança, controle de concorrência e estabilidade, tais como Unix, Linux, Windows 2003 Server, etc;
- **Versatilidade em comunicação:** o servidor pode usar diferentes protocolos de comunicação, tais como TCP/IP, NetBEUI, SPX, etc.

Dentre as características do Cliente destacam-se:

- **Estreita relação com o usuário:** a configuração da máquina, a utilização de softwares, pode ser conforme a necessidade e utilização de cada usuário;
- **Acesso a diversos servidores:** os clientes podem acessar servidores diferentes e de outras configurações;
- **Interface gráfica amigável:** o sistema operacional pode ser um sistema simples, sem muitos recursos adicionais, tais como o Windows, OS/2, etc.

A arquitetura cliente/servidor, mesmo tendo muitas vantagens, tem suas desvantagens, tais como: (i) dificuldades na hora de distribuir e/ou atualizar uma aplicação, caso forem muitas estações; (ii) as regras de negócio não estão separadas nem do cliente e nem do servidor, dificultando assim no momento de fazer uma manutenção nestas regras; (iii) o cliente tem acesso direto aos dados, tendo assim uma menor segurança em relação aos dados, já na arquitetura multicamadas a aplicação servidora gerencia o fluxo de dados entre o cliente e o servidor de dados.

## 2.3 Multicamadas

A arquitetura Multicamadas, também conhecida como *Multitier ou N-Tier*, é uma técnica de desenvolver aplicações, tendo como base a segurança, a escalabilidade, a manutenibilidade, a organização arquitetural, o desempenho e a confiabilidade dos sistemas. O uso desta tecnologia tem sido muito utilizada por empresas que possuem um grande volume de dados e de estações de trabalho, onde fatores como a distribuição da aplicação, a segurança, a escalabilidade, a manutenibilidade, o tráfego de rede e a performance são fatores críticos (CÔRTEZ, 2002).

Neste modelo, a aplicação servidora gerencia o fluxo de dados entre os clientes e o servidor de banco de dados. Ou seja, o servidor de aplicação coordena e processa as requisições e atualizações de múltiplos Clientes. A aplicação cliente não tem acesso aos dados e nem sabe como estes estão armazenados e/ou são mantidos no servidor de dados. Conseqüentemente, o cliente pode focar seu trabalho na interação com o usuário (CÔRTEZ, 2002).

O desenvolvimento em Multicamadas é uma realidade inquestionável e, principalmente, uma tendência. Com ele, pode-se dividir a equipe de desenvolvimento em equipes menores. Uma equipe poderá trabalhar com a camada da interface, sem se preocupar com as validações que serão necessárias (regras de negócios), a qual pode ser executada por outros desenvolvedores que irão programar a camada intermediária. Ainda irão existir desenvolvedores específicos para programar e especificar o banco de dados. Tem-se então uma boa divisão da equipe de desenvolvimento, sem que uma equipe interfira nas tarefas das outras (BATTISTI, 2001).

Quando se fala em desenvolvimento de sistemas em Multicamadas, está se referindo a um modelo de programação que prevê a divisão do programa em três ou mais partes bem definidas e distintas. As três partes principais são as seguintes: **Interface, Regras de Negócio e Persistência** (BATTISTI, 2001; RODRIGUES, 2002. p.8 e 9).

No desenvolvimento de um sistema na arquitetura Multicamadas, deve-se ter uma preocupação constante com a expressão “bem definidas e distintas”. A divisão e não intromissão de uma camada na outra é a característica fundamental desse modelo e a camada mais relevante e importante neste modelo é a de **Regras de Negócio**.

### 2.3.1 Interface

A primeira regra na construção da Interface deve ser a economia e a simplicidade de código. A necessidade de manutenção de um programa é diretamente proporcional à sua complexidade. Portanto, deve-se desenvolver interfaces (janelas, relatórios, etc.) simples e estáveis, já que essa é a parte do programa que deve ser distribuída aos usuários.

O objetivo da interface é acessar os dados localmente ou através da Internet, conectando-se ao servidor de aplicação de qualquer local ou máquina, tendo para isto, um navegador (*Browser*) ou a aplicação cliente instalada. Essa camada também é conhecida como camada Cliente ou Apresentação.

### 2.3.2 Regras de Negócio

As Regras de Negócio (tarefas e regras que governam o processo), também conhecidas como “Lógica”, determinam de que maneira os dados serão utilizados, tendo a função de servir a camada cliente (interface), executando processos em função de suas requisições. Esta camada foi deslocada para o Servidor de Aplicações (camada intermediária). Desta maneira, quando uma regra de negócio for alterada, basta atualizá-la no servidor de aplicações. Após a atualização, todos os usuários passarão a ter acesso a nova versão, sem que seja necessário reinstalar o programa em cada um dos computadores da rede, facilitando a tarefa de manter o sistema atualizado. A “inteligência” do sistema deve se concentrar nessa camada, sendo que todo e qualquer acesso aos dados, deve ser feito através das regras desta camada.

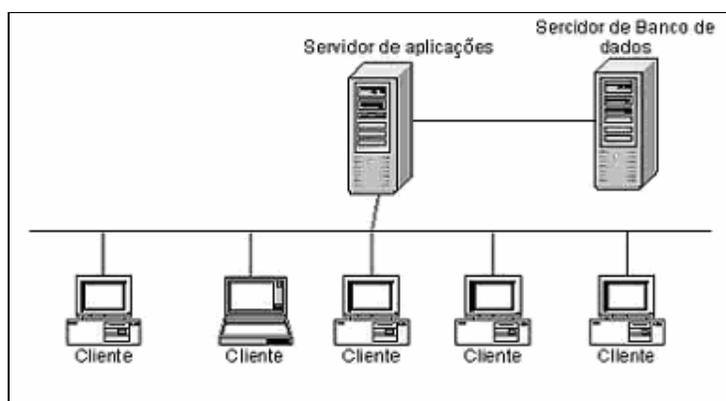
**Servidores de Aplicação** são gerenciadores de alocação de recursos que são responsáveis por gerenciar:

- **Requisições de Clientes:** atendem as solicitações dos clientes, buscando as informações solicitadas no banco de dados e as retornando aos clientes;
- **Alocação de *Threads*** – alocação de fluxos de controles em separado dentro de um sistema, possibilitando que tarefas possam ser executadas simultaneamente;
- **Tolerância à falhas:** quando um servidor de dados estiver com problemas, o servidor de aplicação pode buscar as informações solicitadas pelo cliente em outro servidor de banco de dados;

- **Segurança:** o cliente não tem acesso direto ao banco de dados, somente através do servidor de aplicação;
- **Transações de objetos:** vários objetos diferentes podem ser colocados no contexto de uma única transação, mesmo que esse objeto não faça acesso ao Banco de Dados;
- **Pools:** capacidade de um mesmo objeto conseguir manter algumas instâncias atendendo a dezenas ou até milhares de Clientes;
- **Balanceamento de carga:** quando um servidor fica sobrecarregado, outro servidor pode continuar o processamento.

### 2.3.3 Persistência

A camada de Persistência (mecanismo de armazenamento) fica no servidor do banco de dados, na qual reside toda a informação necessária para o funcionamento da aplicação. Cabe ressaltar, que os dados somente são acessados através do Servidor de Aplicação, e não diretamente pela aplicação do cliente (figura 2.3).



**Figura 2.3 – Estrutura física do modelo em Multicamadas**

Uma das grandes vantagens e a qualidade de um sistema desenvolvido com a arquitetura Multicamadas é a separação da lógica da aplicação em uma camada intermediária separada do Software. A camada de apresentação é relativamente livre de processamento ligado a aplicação. As janelas (apresentação) repassam as solicitações das tarefas para a camada intermediária, a qual se comunica com a camada de armazenamento (figura 2.4) (LARMAN, 2000).

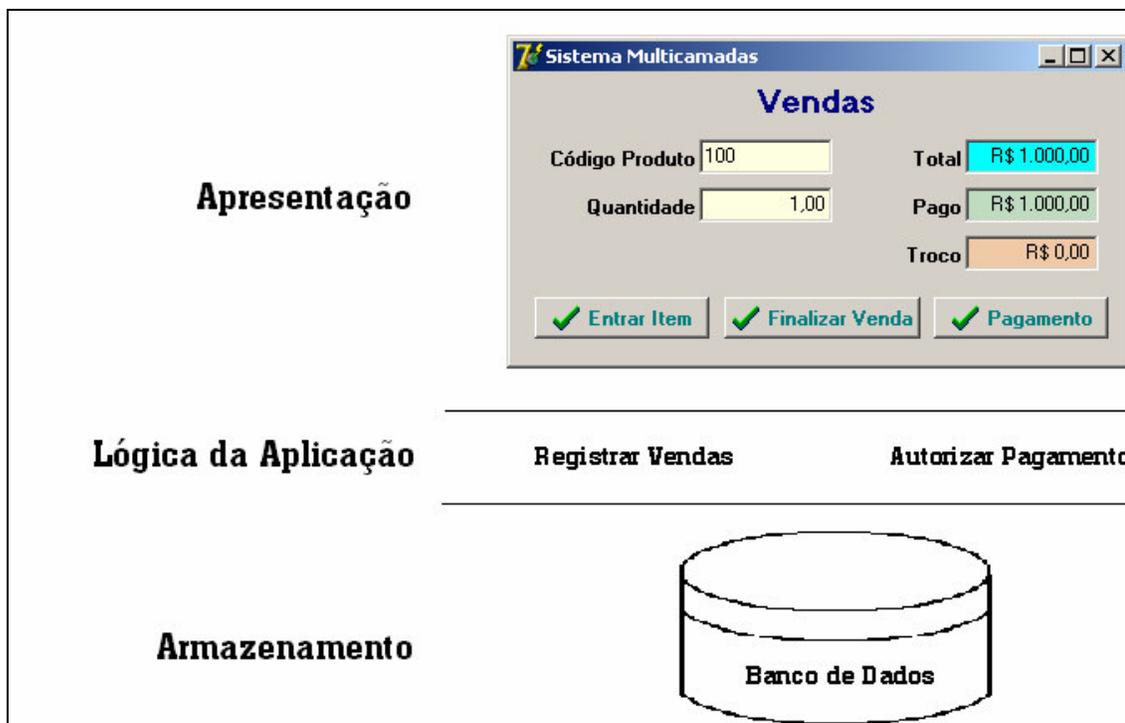
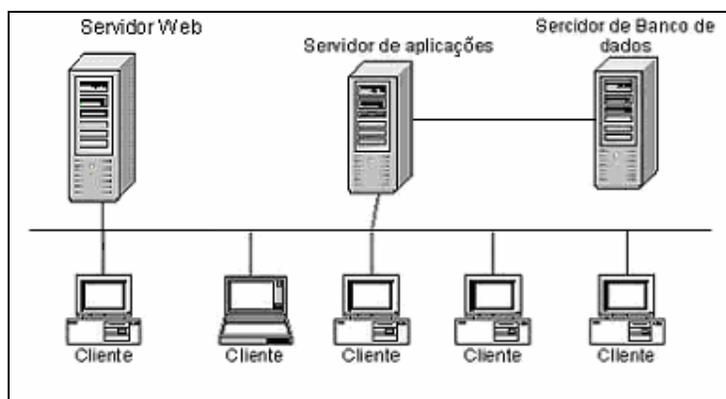


Figura 2.4 – Estrutura lógica do modelo Multicamadas

## 2.4 A Quarta Camada

Como uma evolução do modelo de Três Camadas, surgiu o modelo de Quatro Camadas. O conceito básico do modelo de quatro camadas é a retirada da camada de apresentação do cliente, centralizando-a em um determinado ponto, o qual na maioria dos casos é um Servidor Web. O acesso a aplicação é feito através de um navegador utilizado pelo cliente, como o *Internet Explorer*, o *Netscape Navigator* ou outro navegador qualquer.

A Interface passa para o Servidor Web e esta pode ser composta de páginas HTML, ASP, ou qualquer outra tecnologia capaz de gerar conteúdo para o navegador. Com isso, alterações na interface da aplicação são feitas diretamente no Servidor Web, sendo que estas alterações estarão, automaticamente, disponíveis para todos os clientes (figura 2.5) (BEZERRA, 2002. p. 247-249).



**Figura 2.5 – Estrutura física do modelo em Quatro Camadas**

O Servidor de Aplicação, Servidor Web e Servidor de Banco de dados não precisam, necessariamente, serem servidores distintos, isto é, uma máquina pode fazer o papel de todos os servidores. O conceito de Servidor de Aplicação, Web e Banco de Dados é um conceito relacionado com a função que cada servidor desempenha. Portanto, pode-se ter, em um mesmo equipamento, um Servidor de Aplicações, um servidor Web e um Servidor de Banco de Dados. No entanto, nestes casos, deve-se considerar o desempenho de um modo geral.

É possível acrescentar camadas adicionais e decompor ainda mais as existentes, dependendo até que nível e/ou camada quer se fazer o detalhamento do sistema, tendo sempre como base a organização arquitetural e o desempenho do sistema.

## 2.5 Tecnologia Mista

Com o modelo multicamadas tem-se a opção de usar uma arquitetura multiplataforma, ou melhor, pode-se optar pelo uso de plataformas diferentes em camadas distintas. Sendo assim, será possível utilizar, por exemplo, um servidor em Windows e clientes em Linux, ou vice-versa. Outra possibilidade é fazer com que os sistemas legados (sistemas tradicionais), que tenham sua base de dados em arquivos em um *Mainframe*, possam interagir com clientes através de um servidor de aplicação (BATTISTI, 2001).

A figura 2.6 mostra o uso de uma arquitetura multiplataforma. Neste exemplo tem-se três clientes distintos: (i) Delphi - cliente desenvolvido em Delphi; (ii) Java - cliente desenvolvido para o ambiente Web e (iii) C++ - cliente desenvolvido em C++. O servidor de aplicação pode ser um aplicativo executável (EXE) ou uma DLL e neste exemplo ainda se

tem duas formas de armazenamento, uma como arquivos de um *Mainframe* e a outra como um banco de dados Oracle.

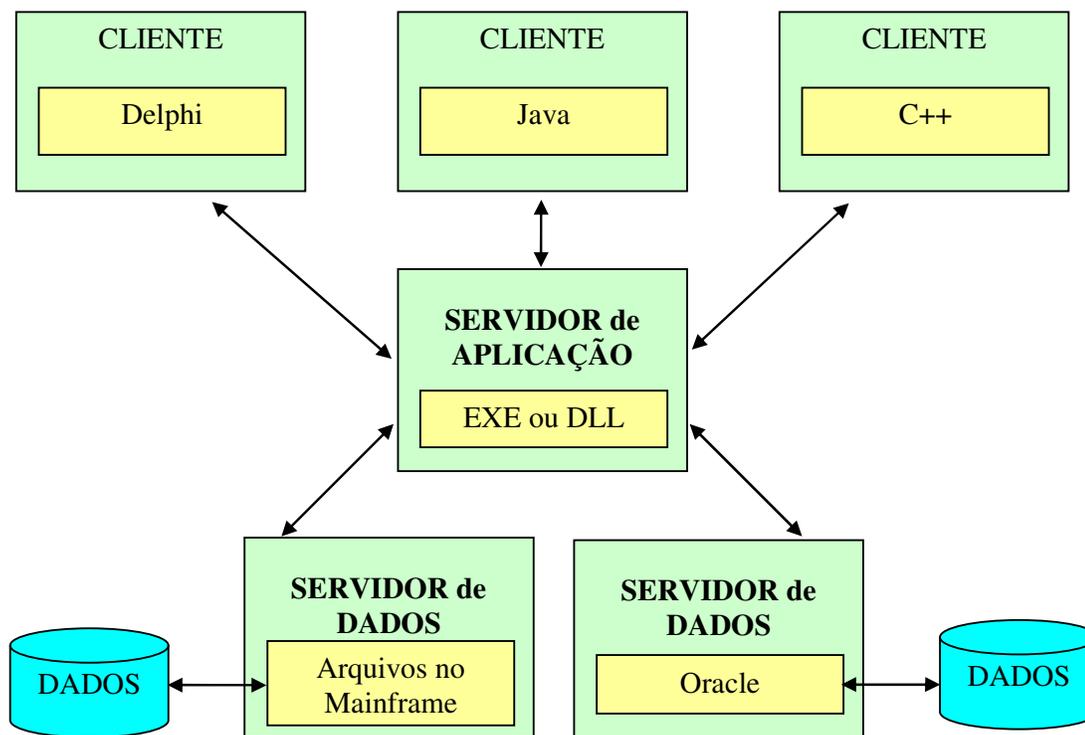


Figura 2.6 – Tecnologia Mista

Resumindo, é possível ter uma aplicação Web nCamadas para consultar o saldo e/ou extratos de uma conta e fazer pagamentos. Um Navegador qualquer pode ser o cliente e através de um servidor de aplicações acessar os dados. Para o usuário não faz diferença se os dados estão em um *Mainframe* ou em um servidor Intel rodando SQL Server, Oracle ou outro banco de dados qualquer (BATTISTI, 2001).

O uso da tecnologia mista ou multiplataforma também é interessante para as equipes de desenvolvimento do projeto, pois uma equipe pode ser especializada em trabalhar com um banco de dados numa plataforma e uma outra equipe de desenvolvimento ser especializada em desenvolver a parte referente ao usuário em outra plataforma (BATTISTI, 2001).

## 2.6 Cliente/Servidor x Multicamadas

Sistemas construídos segundo a arquitetura Cliente/Servidor podem ser encontrados em grande parte das empresas. Muitas destas empresas consideram esta tecnologia satisfatória. Destas afirmações, surge o seguinte questionamento: “qual a necessidade e quais as vantagens de migrar os sistemas da arquitetura Cliente/Servidor para a arquitetura Multicamadas?” (RODRIGUES, 2002. p.9).

Caso esteja-se satisfeito com a arquitetura Cliente/Servidor, não tendo problemas como por exemplo com a manutenibilidade e escalabilidade do sistema, não será necessário fazer a migração do mesmo. Mas, caso o sistema se encontre numa destas situações, este deverá ser revisto e talvez remodelado e reescrito. Uma boa opção será a arquitetura Multicamadas, devido aos benefícios que esta tecnologia oferece, como a manutenibilidade, escalabilidade, segurança e desempenho, entre outras (RODRIGUES, 2002. p.9-14; CÔRTEZ, 2002; PAULI, 2004).

### 2.6.1 Desvantagens do Cliente/Servidor

Apesar de aplicações cliente/servidor serem apropriadas para empresas de pequeno porte devido ao desenvolvimento ser mais simples, tendem a apresentar problemas à medida que cresce o porte da empresa. Dentre estes problemas destacam-se:

- **Dificuldade de distribuição e/ou atualização** – sempre que uma regra é alterada, o sistema deve ser redistribuído, trazendo problemas de controle de versão e distribuição física do aplicativo e seus complementos;
- **Regras de Negócio ficam com os dados** – as Regras de Negócio ficam junto com os dados, fixas a um determinado banco de dados;
- **Queda de performance** – ocorre quando há um acúmulo de regras ou *constraints* no banco de dados, causando o “gargalo de rede” e também o gargalo de I/O da máquina servidora.

### 2.6.2 Vantagens das Multicamadas

Como a integração de tarefa e a atualização constante dos sistemas (regras de negócio) e em diferentes pontos da rede têm afetado de forma prejudicial o desempenho, a segurança e a escalabilidade dos sistemas Cliente/Servidor, a arquitetura de sistemas

Multicamadas surge como uma proposta para minimizar estes problemas. Algumas vantagens desta arquitetura são:

- **Facilidade de distribuição e/ou atualização** – no modelo Multicamadas, apenas um EXE ou uma DLL que se encontra no Servidor de Aplicações necessitam da modificação. Fazendo isso, todas as estações irão acessar a nova Regra de Negócio ou a Regra de Negócio modificada;
- **Encapsulamento da lógica de negócios** – as regras ficam na camada da lógica de negócios, no servidor de aplicação e não fixas a um determinado banco de dados. Sendo assim, estas podem ser reutilizadas em um outro sistema;
- **Melhor performance** – maior performance com a utilização de vários servidores: (i) balanceamento de carga – quando um servidor ficar sobrecarregado, outro servidor de aplicação poderá auxiliar no processamento e (ii) paralelismo – as regras podem ficar em servidores diferentes e/ou em processos diferentes. Por exemplo, as regras de Vendas em um servidor e as regras de Compras em outro e estas processadas em paralelo;
- **Aplicações clientes menores** – aplicações clientes menores, pois o processamento é realizado no nível intermediário;
- **Alocação de desenvolvedores para a construção de camadas específicas** – equipes podem trabalhar exclusivamente para cada camada do sistema, tornando os desenvolvedores especialistas. O desenvolvimento pode ser executado simultaneamente, ou melhor, cada equipe em sua camada em paralelo;
- **Capacidade de segurança aumentada** – como um sistema, não fica centralizado somente em uma ou duas máquinas, quando um servidor “X” parar de funcionar, um servidor “Y” poderá continuar o processamento;
- **Economia de licenças de acesso ao banco de dados** – necessidade de adquirir somente uma licença de acesso aos dados, para o servidor de aplicações, pois somente este terá acesso ao banco de dados. Além disto, o banco de dados tem um número reduzido de conexões para gerenciar, melhorando a performance.

## 2.7 Modelos de Distribuição da Aplicação

De acordo com a estrutura física de uma empresa, as camadas podem ser organizadas de diferentes formas, cada uma tendo suas vantagens e desvantagens. Para esta distribuição, sempre deve ser considerada a estrutura física da organização e de que forma o sistema como um todo será utilizado pelos usuários (KALSING, 2003).

### A) Negócio e Dados no mesmo Servidor

As regras de negócio (serviço de negócio) e a camada da persistência (serviço de dados) podem ficar no mesmo servidor. Desta forma os clientes, através do serviço de apresentação acessam somente o serviço de negócio e este busca os dados, através do serviço de dados, retornando-os aos clientes conforme a solicitação (figura 2.7).

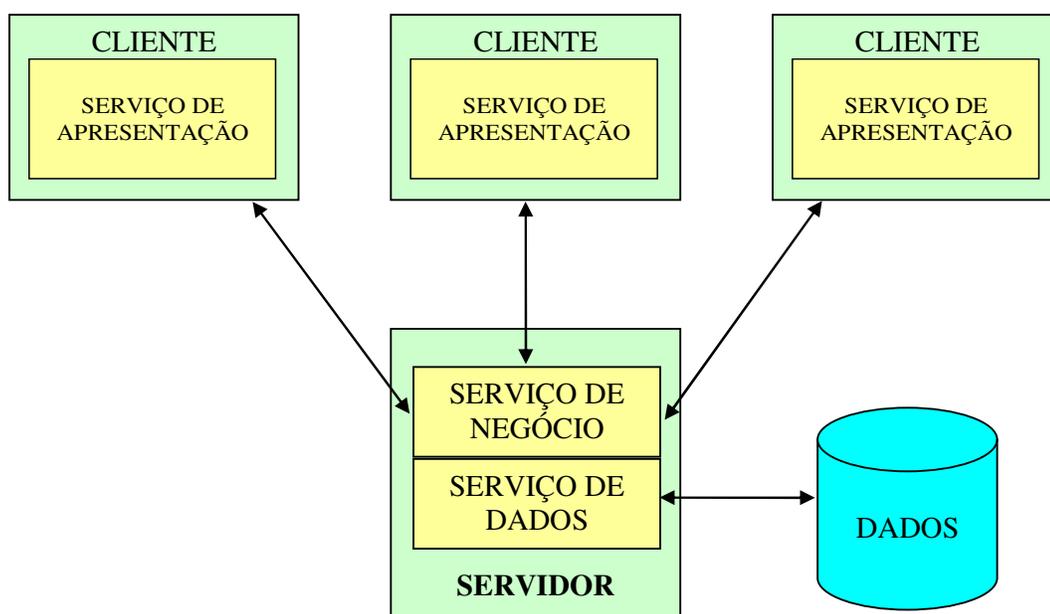


Figura 2.7 – Negócio e Dados no mesmo Servidor

## B) Negócio e Dados em Servidores diferentes

Para não sobrecarregar somente um servidor, pode-se fazer o uso de dois servidores diferentes, um para o serviço de negócio, o qual será o servidor de aplicação e outro para o serviço de dados, o qual será o servidor de dados. Da mesma forma como a distribuição anterior, os clientes através do serviço de apresentação, terão acesso somente ao serviço de negócio (figura 2.8).

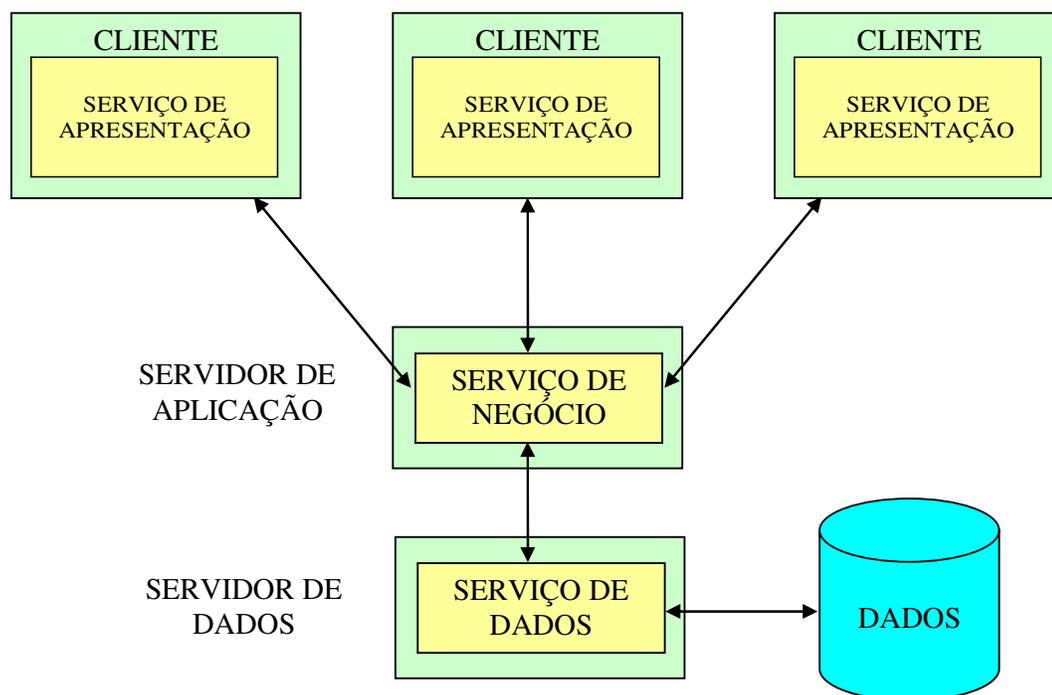


Figura 2.8 – Negócio e Dados em Servidores diferentes

### C) Negócio e Dados em dois Servidores idênticos

Para se ter uma alta disponibilidade e distribuição de processamento, ou melhor, processamento paralelo, pode-se fazer o uso de servidores idênticos. Sendo assim, o processamento das solicitações dos clientes será feito pelo servidor que estiver menos sobrecarregado (figura 2.9).

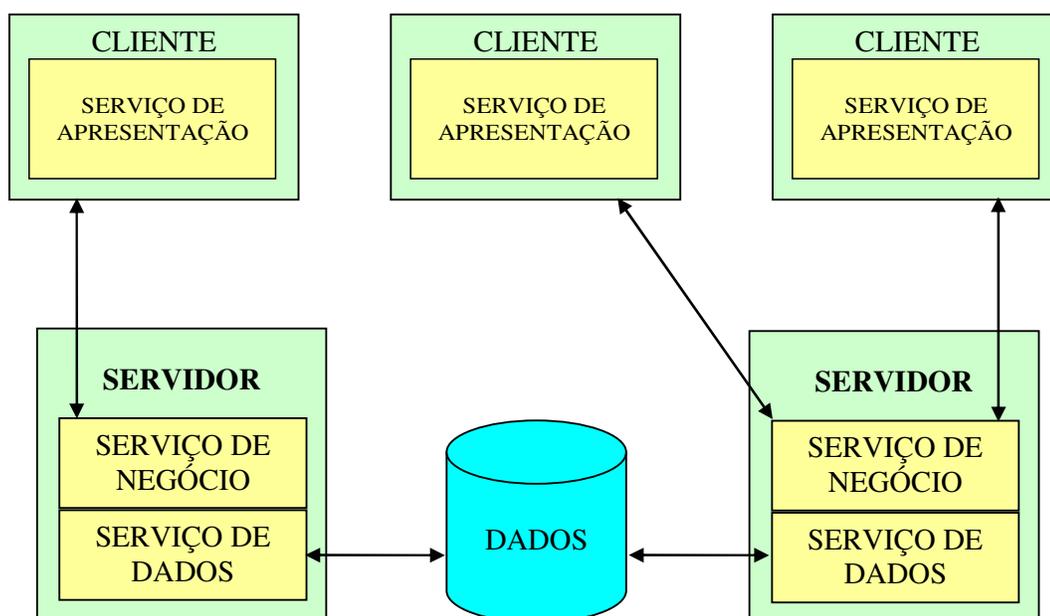


Figura 2.9 – Negócio e Dados em dois Servidores idênticos

**D) Negócio e Dados separados utilizando um Banco de Dados particionado**

Da mesma forma como a distribuição anterior, para se ter alta disponibilidade e distribuição de processamento, pode-se também fazer o uso de um serviço de negócio e dois serviços de dados e tendo assim o banco de dados particionado. Esta distribuição é muito útil para quando o banco de dados é muito grande (figura 2.10).

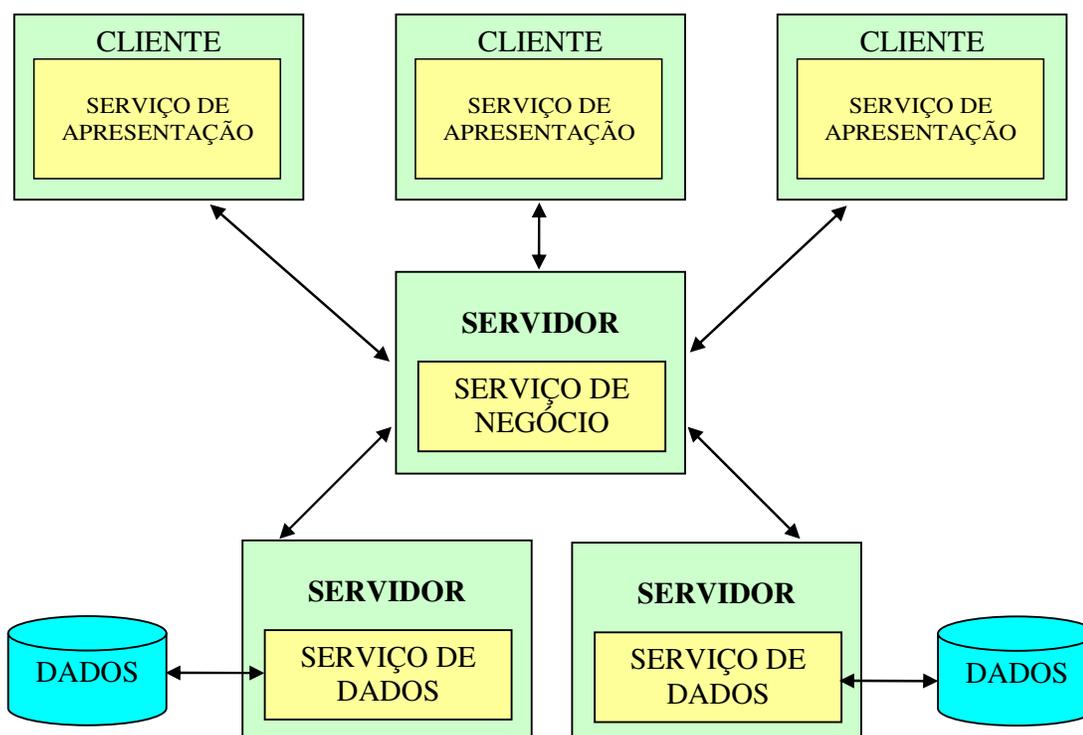


Figura 2.10 – Negócio e Dados utilizando um Banco de Dados particionado

### 3 PROTOCOLOS DE COMUNICAÇÃO

O transporte de dados entre as camadas de uma aplicação é feito através da utilização de protocolos de comunicação, também conhecido como mecanismo de RPC (*Remote Procedure Call*). Existem diversos e diferentes tipos de protocolos de comunicação, cada um com seus padrões<sup>5</sup> definidos e tendo suas vantagens e desvantagens (CANTÚ, 2003). Atualmente existem vários protocolos de comunicação, sendo que os mais conhecidos e usados são o CORBA da OMG, o DCOM da Microsoft e o Java RMI da Sun (ARAUJO, 2001).

#### 3.1 CORBA

CORBA (*Common Object Request Broker Architecture*) é uma arquitetura genérica e portátil para objetos distribuídos, tendo como finalidade, facilitar a computação distribuída de objetos. O CORBA propõe total independência das plataformas de hardware e das linguagens de programação (RODRIGUES, 2002. p.28).

CORBA é uma padrão aberto, padronizado pelo consórcio OMG (*Object Management Group*) composto por mais de 800 representantes do setor. O OMG especifica o que a arquitetura CORBA fará e, até certo ponto, como ela o fará. Fora isso, cada fornecedor de arquitetura CORBA é livre para criar sua própria implementação e método compatível com a especificação CORBA (TEIXEIRA, 2002).

---

<sup>5</sup> Os padrões de protocolos são normas e especificações técnicas implementadas, testadas e documentadas por organizações e ou consórcios de empresas.

### 3.1.1 Características

Destacam-se como características do CORBA (TEIXEIRA, 2002):

- **CORBA é uma técnica orientada a objeto** – todo servidor CORBA publica uma interface que lista os métodos e os tipos de dados que ele aceita. Os detalhes da implementação ficam escondidos de quem chama;
- **Transparência de Local** – os objetos podem estar localizados em qualquer lugar. Quando uma aplicação cliente CORBA chama um objeto servidor, ela não sabe onde o servidor reside, mas recebe uma imagem da aplicação servidora;
- **Independência da linguagem de programação** – os objetos podem ser escritos em diversas linguagens diferentes. Java e C++ são os líderes, mas o Delphi está obtendo uma boa aceitação devido aos recursos disponíveis no produto. Os objetos CORBA interagem uns com os outros por meio de suas interfaces publicadas. Cada objeto servidor precisa ser compatível com sua definição de interface;
- **Múltiplas plataformas e Sistemas Operacionais** – existem implementações CORBA para diferentes plataformas e Sistemas Operacionais. Por exemplo, uma distribuição pode usar Java na camada de persistência e o Delphi na camada intermediária ou no lado do Cliente. Os desenvolvedores, tendo como base as últimas tecnologias e ferramentas, podem escrever aplicações robustas, que aproveitam os sistemas legados e apresentem informações para os usuários finais com muitos recursos.

### 3.1.2 Arquitetura

A Figura 3.1 mostra um diagrama de blocos da arquitetura CORBA. A parte comum para Cliente e Servidor é o ORB, o qual trata de toda a comunicação entre os objetos. Ele faz isto usando o *Internet Inter-ORB Protocol* (IIOP), que se encontra uma camada acima do TCP/IP. Isso garante a entrega e o uso de mensagens confiável de ponta a ponta. Além de cuidar de todo o tráfego de mensagens, o ORB também corrige variações de plataforma.

O *Stub* converte a chamada do cliente e a envia pela rede utilizando o ORB. Depois, o ORB do servidor recebe a requisição do cliente e a envia para o esqueleto (*skeleton*). O

esqueleto contém o resultado da requisição através da implementação e a resposta é enviada ao cliente utilizando o ORB (TEIXEIRA, 2002).

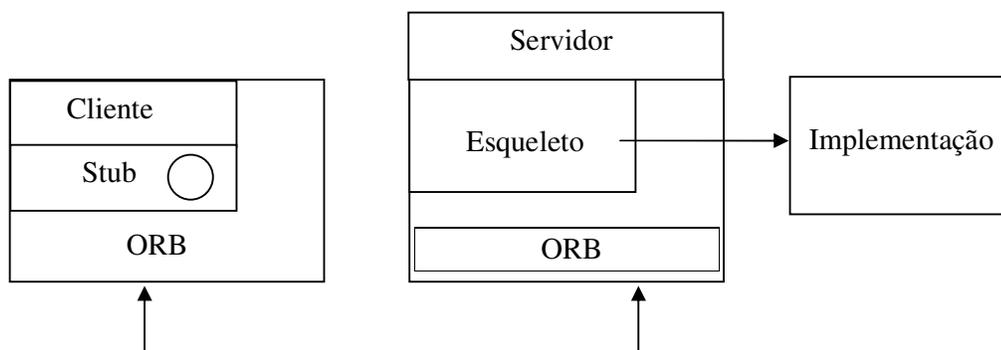


Figura 3.1 – A arquitetura CORBA

### 3.1.3 Interfaces

Todos os objetos CORBA são descritos por suas Interfaces. Esse é um projeto puramente orientado a objeto. Uma Aplicação Servidora publicará declarações de tipo, interfaces e métodos específicos, que qualquer Cliente poderá chamar. Quando essas interfaces são publicadas, elas se tornam imutáveis. Para adicionar outros recursos a um objeto já publicado, a melhor técnica é derivar um novo servidor a partir do antigo e melhorar o novo objeto. Desse modo uma nova interface pode ser publicada sem o surgimento de incompatibilidades para as aplicações distribuídas.

Para descrever as Interfaces, o consórcio da OMG publicou uma linguagem de definição de Interface (*Interface Definition Language – IDL*). A IDL é independente da linguagem de programação, mas é parecida com a linguagem C ou Java.

## 3.2 COM

COM (*Component Object Model*) foi desenvolvido e padronizado pela Microsoft em 1993, tendo como função fornecer suporte a diferentes módulos e/ou partes de softwares que necessitam se comunicar. COM é uma especificação de Interfaces binárias para comunicação e pode ser desenvolvido em linguagens distintas, tais como: Delphi, C++, Visual Basic, Java, etc. (LÖWY, 2001, p.1).

O objeto COM não toma conhecimento em que linguagem foi desenvolvido, mas o que importa, é o acesso a este objeto através de interfaces. É a interface que fornece suporte à chamada do objeto escrito em qualquer linguagem. Na construção de um objeto COM, é necessário entender como as interfaces trabalham. Já o usuário da interface não precisa conhecer os detalhes de implementação. Este isolamento de interface e implementação é crucial para objetos COM. Isolando a interface de sua implementação, é possível construir componentes que podem ser substituídos e reutilizados. Isto simplifica e multiplica a utilidade do objeto (RODRIGUES, 2002. p.17 e 18).

A interface de comunicação pode ser escrita na forma de Biblioteca de Ligação Dinâmica (DLL) ou programas executáveis (EXE). Sendo que, se for uma DLL, o mesmo é chamado de “*In-Process*”, e se for um EXE, é chamado de “*Out-of-Process*” (CÔRTEZ, 2002).

***In-Process*** possui todas as características de uma DLL, sendo assim, a mesma é executada na mesma área de memória do processo chamador, isto é, o seu EXE (aplicação) ou até mesmo uma outra DLL.

***Out-of-Process*** é um EXE que possui uma área independente de memória, não fazendo parte da área de memória do processo chamador.

### 3.2.1 DCOM

DCOM (*Distributed Component Object Model*) é a versão distribuída no modelo COM, tendo suas raízes baseadas no COM, OLE e ActiveX. Deve-se considerar COM e DCOM como sendo uma única tecnologia. Se o objeto necessitar usar somente COM, o mesmo será usado, mas se o objeto estiver em uma máquina distinta, será utilizado o DCOM (RODRIGUES, 2002. p.20).

### 3.2.2 MTS

MTS (*Microsoft Transaction Server*) é o servidor de transações da Microsoft, uma extensão do COM e que incorpora vários recursos ao DCOM. Em outras palavras, MTS é uma evolução do DCOM.

Objetos MTS são objetos COM com suporte a transações para Banco de Dados. O MTS é parte integrante do Windows 98 e Windows NT, se encontra no *IIS (Internet Information Services)* e é necessário para a criação de sistemas multicamadas com acesso a dados.

### 3.2.3 COM+

COM+ (*Component Object Model Plus*) foi criado em 1996, sendo o protocolo que contém todas as características do DCOM integrado com o *Microsoft Transaction Server* (MTS). Além disto, possui uma série de recursos adicionais, que são (RODRIGUES, 2002. p.22 e 23):

- **Pooling** – é a capacidade de um mesmo objeto conseguir manter algumas instâncias atendendo a dezenas ou até milhares de Clientes. É desta forma que o COM+ é considerado escalonável;
- **Escalabilidade** – pode-se aumentar drasticamente o número de conexões concorrentes ao objeto e o COM+ tende a não perder a sua performance, pois estará fazendo o uso do *Pooling*;
- **Gerenciamento** – é o console de gerenciamento (*Component Services*) de objetos do COM+. Com este se fará a instalação, a exportação e a administração dos objetos COM+;
- **Transações** – o mecanismo responsável pelas transações no COM+ é o MSDTC, que também se encontra no *Component Services*. Com o COM+, pode-se realizar transações em objetos e em Banco de Dados. As transações de objetos permitem que vários objetos diferentes sejam colocados no contexto de uma única transação, mesmo que esse objeto não faça acesso ao Banco de Dados;
- **Segurança** – o objeto COM+ tem sua segurança baseada nas seguintes medidas: (i) atribuições (*roles*) - o gerenciamento de contas é baseada em categorias ou grupos de usuários e não por usuários individuais; (ii) segurança programática - controle através do acesso a métodos específicos, verificando se o usuário pertence a uma atribuição (*role*) específica;

### 3.3 SOAP

SOAP (*Simple Object Access Protocol*) é uma das mais recentes tecnologias em relação aos protocolos de comunicação. Este protocolo é um modo de usar a infra-estrutura da Internet existente, para permitir que as aplicações se comuniquem diretamente entre si, sem serem bloqueadas por *Firewalls*<sup>6</sup>.

SOAP foi definido em XML (*Extensible Markup Language*) e quando invocam-se métodos através de RPC (*Remote Procedure Call*), estes também são codificados em XML. O transporte das mensagens é feito através de HTTP que o transforma em um protocolo leve, e ainda, elimina problemas com o uso do *Firewall* no servidor. O SOAP tende a facilitar o desenvolvimento de sistemas e a conversação entre sistemas distribuídos, já que o mesmo é feito em XML (RODRIGUES, 2002. p.30). As principais vantagens do SOAP são:

- **Simples de implementar** – o protocolo SOAP é fácil de usar e implementar;
- **Padrão da indústria** – criado por um consórcio da qual a Microsoft faz parte e adotado pela W3C (*World Wide Web Consortium*)<sup>7</sup> e por várias outras empresas conceituadas;
- **Usa os mesmos padrões da Web** - a comunicação é feita via HTTP com pacotes virtualmente idênticos; os protocolos de autenticação e encriptação são os mesmos e normalmente são implementados pelo próprio Servidor Web;
- **Atravessa Firewalls e Roteadores** – estes “pensam” que o SOAP é uma comunicação HTTP;
- **Descrição em XML** – no protocolo SOAP tanto os dados como as funções são descritas em XML;
- **Portabilidade** – o SOAP é independente de sistema operacional e plataforma de hardware.

O protocolo SOAP pode ser facilmente implementado em qualquer ambiente de programação. Atualmente existem diversos pacotes de desenvolvimento SOAP para diferentes sistemas operacionais e linguagens de alto nível, tais como: Delphi, .NET, etc.

---

<sup>6</sup> *Firewall* é um dispositivo de segurança que monitora o tráfego de informação entre uma rede de computadores e a Internet, impedindo o acesso de usuários não autorizados ou a entrada de dados sem a prévia permissão.

<sup>7</sup> W3C é um consórcio de Empresas as quais definem os padrões e as especificações em relação à Internet. Disponível em <<http://www.w3.org/TR/SOAP/>>. Acesso em 01 jun. 2004.

Inclusive, o SOAP é uma parte importante da arquitetura .NET da Microsoft<sup>8</sup> (SANT'ANNA, 2003).

### 3.4 Java RMI

Java RMI (*Remote Method Invocation*) é um mecanismo que permite um objeto invocar um método que está em outro espaço de endereçamento. Este outro espaço de endereçamento pode ser na mesma máquina ou em uma máquina diferente. O RMI é um mecanismo de RPC para programação orientada a objetos.

Recentemente, o RMI se tornou compatível com o CORBA, ou melhor, foi implementada uma nova versão de RMI, baseada no IIOP (protocolo do CORBA), chamada de RMI/IIOP. Esta versão usa o protocolo IIOP (*Internet Inter-ORB Protocol*) como *background* da comunicação do RMI (LIMA, 2004)<sup>9</sup>.

#### 3.4.1 Comunicação de Objetos Distribuídos

Quando dois objetos em espaços de endereçamento diferentes se comunicam, estes precisam de intermediários para realizar a transmissão. Este intermediários são o *Stub* e o *Skeleton*.

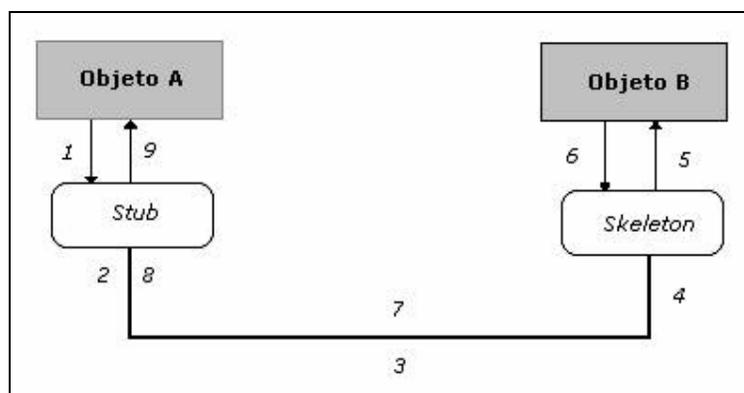


Figura 3.2 – Comunicação de Objetos Distribuídos

Em relação a figura 3.2, o objeto “A” não conhece a implementação do objeto “B”, conhece apenas uma interface remota. A interface remota define quais métodos podem ser

<sup>8</sup> Disponível em <<http://msdn.microsoft.com/soap/default.asp>>. Acesso em 01 jun. 2004.

<sup>9</sup> Disponível em <<http://www.jspbrasil.com.br>>. Acesso em 03 jun. 2004.

chamados remotamente. Tanto o objeto “B” quanto o *Stub* implementam a interface remota. Assim, a seqüência de passos para realizar a comunicação de um objeto distribuído é a seguinte:

1. O objeto “A” faz a chamada do método desejado no *Stub*.
2. O *Stub* converte a chamada em protocolo RMI e envia pela rede.
3. A requisição é enviada pela rede.
4. O *Skeleton* recebe a requisição em RMI e reconhece qual método deve ser chamado.
5. Chama o método desejado no objeto “B”.
6. Recebe o retorno do método.
7. Envia o objeto de retorno para o Cliente.
8. O *Stub* recebe o retorno.
9. O *Stub* repassa o retorno para o objeto “A”.

Esse é o processo que permite que objeto “A” se comunique com o objeto “B” mesmo eles estando em máquinas diferentes. Os *Stubs* e *Skeletons* são objetos gerados automaticamente por alguma ferramenta e sem eles, não seria possível a comunicação entre objetos.

### **3.5 Considerações Finais**

Como pode ser observado, os protocolos trabalham com soluções similares. A diferença entre as tecnologias está em seus recursos bem como em sua complexidade. Cada tecnologia opera de forma diferente, ou seja, uma aplicação que usa DCOM não pode se comunicar com outra que usa RMI.

Com o Delphi (ferramenta de programação visual) pode-se trabalhar com os seguintes protocolos: CORBA, COM, DCOM, MTS, COM+ ou SOAP. Dentre estes, o protocolo SOAP tem diversas vantagens sobre as outras formas de chamar funções remotamente, e ainda, é de fácil uso e implementação. O SOAP também usa os mesmos padrões da Web e atravessa os *Firewalls*, pois a comunicação é feita via HTTP.

Sendo assim, a escolha pela ferramenta do Delphi, com a utilização do protocolo SOAP, para o desenvolvimento de dois módulos do Sistema de ERP, nas próximas etapas, concretizou-se pelos motivos acima mencionados.

## 4 MULTICAMADAS NO DELPHI

Delphi é considerado uma ferramenta RAD (*Rapid Application Development*). Caracteriza-se como um ambiente de programação visual e orientado a objeto. O Delphi tem várias ferramentas de suporte ao desenvolvimento tais como: componetes, *templates* e *experts* de aplicações e formulários, que aumentam muito a produtividade, facilitando a programação e codificação da aplicação (CANTÚ, 1998. p.3 e 4).

As próximas seções apresentam os recursos e componentes usados no Delphi para o desenvolvimento de aplicações Multicamadas.

### 4.1 DataSetProvider

O DataSetProvider é um dos principais componentes da tecnologia Multicamadas e também pode ser utilizado na tecnologia Cliente/Servidor. A classe TDataSetProvider disponibiliza os dados do Servidor de Banco de Dados para a aplicação Cliente, e a mesma utiliza (por intermédio do ClientDataSet) o DataSetProvider para atualizar os dados no Servidor de Banco de Dados.

Resumindo, o componente DataSetProvider serve de ponte entre o cliente e o servidor. Os dados providos são chamados de pacotes (*Packets* ou *Data Packet*). Um pacote de dados é um bloco de dados que trafega do servidor de aplicação para o cliente ou do cliente para o servidor. Esta comunicação, entre Servidor e Cliente, é feita através dos protocolos de comunicação (RODRIGUES, 2002. p.125-127).

As duas principais operações que o DataSetProvider e o ClientDataSet utilizam em conjunto são:

- **Providing** - quando se inicia o processo de abertura de um conjunto de dados no lado cliente, o componente ClientDataSet estabelece uma conexão com o seu provedor (provedor) - que neste caso é um DataSetProvider - e solicita o *data packet*, ou melhor, solicita os dados do servidor;
- **Resolving** - quando o ClientDataSet recebe o *data packet* do provider, o mesmo está pronto para realizar qualquer tipo de operação (*insert/append*, *delete* ou *edit*) nos registros retornados (CÔRTEZ, 2002).

A figura 4.1 mostra os componentes envolvidos neste processo.

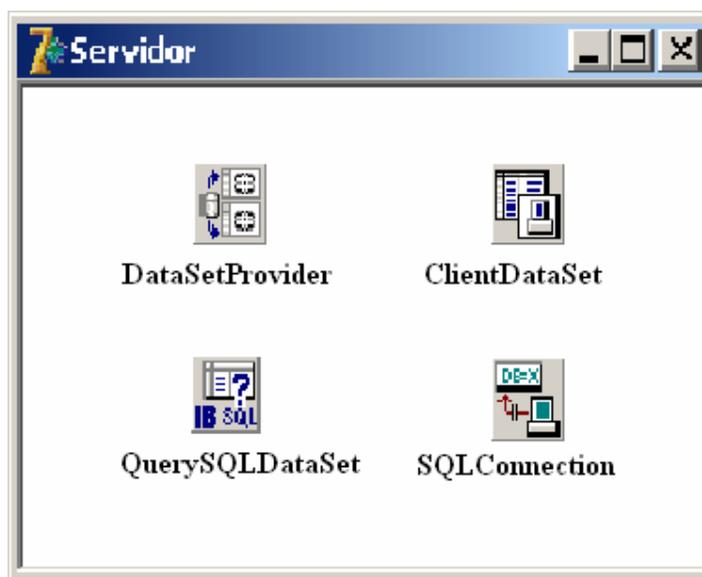


Figura 4.1 – Os Componentes DataSetProvider e ClientDataSet

## 4.2 DataSnap

DataSnap é um conjunto de componentes do Delphi que oferece todos os recursos para a implementação de aplicações Multicamadas. Nas versões anteriores do Delphi estes componentes eram conhecidos por MIDAS (*Middle-tier Distributed Applications Services*). Além dos componentes DataSnap, os componentes para implementação de WebServices disponibilizam uma série de opções de conexão, com os mais variados protocolos, tais como: TCP/IP, COM, SOAP, etc. (figura 4.2) (RODRIGUES, 2002. p.219 e 220).

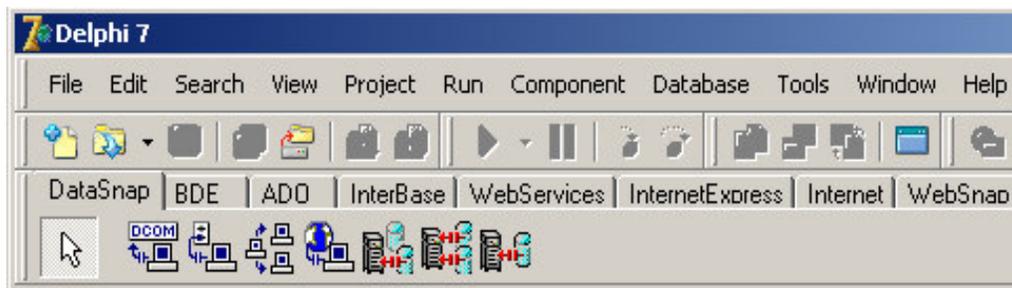


Figura 4.2 – Componentes DataSnap do Delphi

A figura 4.3 mostra a arquitetura de uma aplicação multicamadas baseada em DataSnap. No Servidor de Aplicação, tem-se a interface *AppServer*, a qual fará o “meio-de-campo” entre a aplicação Cliente e o Banco de Dados (PAULI, 2004. p.27).



Figura 4.3 – Arquitetura DataSnap para aplicações Multicamadas

*IAppServer* é a interface definida pela Borland para a comunicação entre um Cliente e um Servidor de Aplicações em uma arquitetura DataSnap que utiliza o protocolo COM ou *Sockets*.

Utilizando o protocolo COM+, o *AppServer* será uma *ActiveX Library*, ou melhor, será uma *.dll* que deve ser registrada no catálogo do COM+. Caso seja usado *SOAP*, o servidor de aplicações pode ser um *.exe* (executável) ou uma *.dll* e um destes será hospedado em um Servidor *Web*, podendo ser o IIS ou o Apache. A interface usada no *SOAP* será a *IAppServerSOAP*.

### 4.3 Web Services

*Web Services* é um serviço Web, ou melhor uma aplicação lógica, programável, que torna compatíveis entre si os mais diferentes aplicativos, independentemente do sistema operacional, permitindo a comunicação e intercâmbio de dados entre diferentes redes

corporativas. A interoperabilidade entre redes distintas é possível pelo uso de padrões como o XML para a troca de mensagens e o SOAP para recebimento e envio dessas mensagens. Ainda em desenvolvimento, essa tecnologia promete uma drástica redução de custos na comunicação e troca de informação entre as empresas (TODD, 2003. p. 327 e 328).

No Delphi, na paleta de componentes *WebServices* tem-se os recursos e os componentes que fazem o uso do protocolo SOAP (*Simple Object Access Protocol*) para fazer a comunicação entre o Cliente e o Servidor de uma aplicação.

O uso do protocolo SOAP tem a vantagem de ser definido em XML. Sendo assim, as chamadas às procedures remotas (RPC) são codificadas em XML e para o transporte das mensagens é usado o HTTP, que além de tornar o SOAP um protocolo leve, elimina problemas em relação a *firewalls*. Isto porque o SOAP usa o HTTP como camada de transporte operando na porta 80, que na maioria dos casos está liberada pelo *firewall* (W3C-*World Wide Web Consortium*)<sup>10</sup>.

#### 4.3.1 Implementando um Servidor com Web Services em Delphi

Para implementar um Servidor com *Web Services*, em primeiro lugar, deve-se criar um projeto *WebServices* e o tipo de aplicação deve ser um CGI (*Common Gateway interface*)<sup>11</sup>, o qual será executado pelo servidor Web IIS (*Internet Information Services*). O projeto gerado é um *Web Server Application*, no qual foram criados os componentes fundamentais para implementação do servidor, que são (figura 4.4):

- **THTTPSoapDispatcher** - este componente atua como despachante recebendo as mensagens recebidas e encaminhando-as para o objeto especificado em sua propriedade “Dispatcher” para que seja decodificada. Ele registra-se automaticamente junto ao *Web Module* como um *auto-dispatching object*. Sendo assim, não é necessário a criação de “Actions” para direcionar as requisições para o THTTPSoapDispatcher, ele recebe todas as solicitações automaticamente;
- **THTTPSoapPascalInvoker** - este componente recebe a mensagem vinda do THTTPSoapDispatcher e a interpreta, cuidando para que seja disparado o método

---

<sup>10</sup> W3C é um consórcio de Empresas as quais definem os padrões e as especificações em relação à Internet. Disponível em <<http://www.w3.org/TR/SOAP/>>. Acesso em 01 jun. 2004.

<sup>11</sup> CGI é um protocolo para a programação de servidores Web, permitindo a geração de páginas HTML dinâmicas.

correspondente à solicitação. Além disto, codifica o retorno do método para o padrão SOAP/XML;

- **TWSDLHTMLPublish** - este componente é responsável por publicar todas as informações registradas pelo *Web Service*.



Figura 4.4 – Criação de um WebModule com o WebServices

Todo servidor *Web Services* deve publicar informações sobre si mesmo no padrão WSDL (*Web Service Description Language*)<sup>12</sup>. No Delphi isso é feito automaticamente pela simples inclusão e configuração do componente **TWSDLHTMLPublish** no projeto do servidor. Essas informações estão acessíveis, executando o Servidor localmente sobre o IIS. Sendo assim, tem-se o acesso a estas informações com o seguinte endereço: <http://localhost/cgi-bin/WebServices.exe/wsdl>. Depois de feito isto, tem-se disponível todas as interfaces implementadas pelo Servidor bem como um link para a descrição WSDL de cada interface.

O segundo passo na implementação de um *Web Services* é a definição das interfaces (permitem o relacionamento de um objeto com o outro) que serão publicadas e usadas pelos seus clientes. Estas interfaces devem ser herdadas de *IInvokable*. Uma interface é definida do tipo “Invokable” para que esta interface seja compilada com a informação em *run-time* (em tempo de execução). O trecho de código apresentado na figura 4.5 mostra a definição de uma interface simples, para as operações aritméticas básicas (VASCONCELOS, 2004)<sup>13</sup>. Esta interface será usada pelos Clientes para solicitar serviços junto ao Servidor.

<sup>12</sup> WSDL é formato de linguagem da descrição de Web Services baseada em XML, padronizada pelo W3C Disponível em <<http://www.w3.org/>>. Acesso em 13 jun. 2004.

<sup>13</sup> Disponível em <[http://www.linhadecodigo.com.br/artigos.asp?id\\_ac=3&pag=1](http://www.linhadecodigo.com.br/artigos.asp?id_ac=3&pag=1)>. Acesso em 13 jun. 2004.

```
unit IMathIntf;  
  
interface  
  
type  
IMath = interface(IInvokable)  
    function Soma(X, Y : Double): Double; stdcall;  
    function Dife(X, Y : Double): Double; stdcall;  
    function Mult(X, Y : Double): Double; stdcall;  
    function Divi(X, Y : Double): Double; stdcall;  
end;  
  
implementation  
    uses InvokeRegistry;  
  
initialization  
    InvRegistry.RegisterInterface(TypeInfo(IMath));  
  
end.
```

**Figura 4.5 – Exemplo da definição de uma Interface**

A figura 4.6 apresenta o código da implementação da Interface Invokable (TInvokableClass) definida na figura 4.5.

```
unit IMathImpl;

interface

Uses InvokeRegistry, IMathIntf;

type
TMath = class(TInvokableclass, IMath)
public
function Soma(X: Double; Y: Double): Double; stdcall;
function Dife(X: Double; Y: Double): Double; stdcall;
function Mult(X: Double; Y: Double): Double; stdcall;
function Divi(X: Double; Y: Double): Double; stdcall;
end;

implementation

{ TMath }

function TMath.Soma(X, Y: Double): Double;
begin
Result := X + Y;
end;

function TMath.Dife(X, Y: Double): Double;
begin
Result := X - Y;
end;

function TMath.Mult(X, Y: Double): Double;
begin
Result := X * Y;
end;

function TMath.Divi(X, Y: Double): Double;
begin
Result := X / Y;
end;

initialization
InvRegistry.RegisterInvokableClass(TMath);

end.
```

**Figura 4.6 – Exemplo da implementação de uma Interface**

Depois de escrito este código, tem-se implementado um Servidor com suporte a *Web Services*, completo e funcional, através do protocolo SOAP. Logo após, pode-se passar para a implementação do Cliente que irá utiliza-lo.

### 4.3.2 Implementando Clientes com Web Services

Esta seção apresenta a implementação dos Clientes com *Web Services*. Criando uma aplicação simples e com a interface conforme a figura 4.7, tem-se a aplicação Cliente. Nesta interface se dará a entrada de dois valores (X e Y). Sobre estes valores se fará uma das quatro operações básicas (processadas pelo Servidor) e o resultado será exibido ao lado.

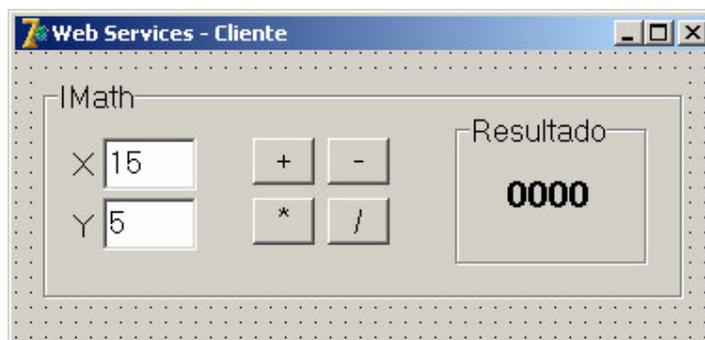


Figura 4.7 – Criação de um Cliente com o WebServices

Com a construção da interface do Cliente e tendo o Servidor já executando e disponível na Web, já se pode requisitar os serviços remotos do *Web Services*.

#### A) Métodos Remotos

Com o componente *THTTTPRIO* se consegue obter uma referência válida de uma interface registrada. Quando se faz um *type cast* (mudança de tipo) desse objeto para uma interface específica, ele gera uma tabela de métodos em memória que é usada quando é feita uma chamada a um método específico.

Para usar esse componente, é necessário configurá-lo, o que pode ser feito de duas maneiras diferentes, dependendo de como o Servidor for implementado:

- **Servidor com Delphi** – neste caso, só é preciso configurar a propriedade URL, cujo valor é gerado no registro “Invokable” da interface;

- **Servidor em outra Linguagem** – devem ser configuradas as propriedades WSDLLocation, Service e Port através do “Object Inspector”, conforme mostra a figura 4.8.



**Figura 4.8 – Configuração do componente THTTTPRIO**

Configurado o componente THTTTPRIO, deve-se partir para a codificação do Cliente, ou melhor condicionar as funções e eventos dos Botões (Soma, Subtração, Multiplicação e Divisão), assim como será visto do Botão “Soma” no código da figura 4.9.

```

procedure TfrmMain.SpeedSomaClick(Sender: TObject);
var
  Im : IMath;
  A, B : Double;
begin
  Im := HRMath as IMath;
  A := StrToFloat(EdtX.Text);
  B := StrToFloat(EdtY.Text);
  LblResult.Caption := FloatToStr(Im.Soma(A, B));
end;

```

**Figura 4.9 – Código do Botão Soma**

Percebe-se que o uso dele é bem simples. Basta declarar uma variável do tipo da interface desejada e atribuir a ela o *type cast* do componente. Depois de feito isso, pode-se usá-lo normalmente como se fosse um objeto. Não é necessário a liberação de memória

correspondente à referência da interface, pois ela é liberada automaticamente quando a variável sai de escopo. Esse procedimento deve ser repetido para todos os métodos que se deseja requisitar.

#### B) TSoapConnection

O componente TSoapConnection é usado por uma aplicação cliente que deseja conectar-se ao servidor de aplicação, implementado como um *Web Services*. Com ele tem-se a possibilidade de estabelecer a conexão entre Servidor e Cliente e obter a *IAppServer* do Servidor, implementado num *RemoteDataModule*, a partir da qual tem-se acesso aos *providers* existentes nele.

Esse componente traz algumas vantagens em relação aos outros existentes que desempenham funções parecidas. Dentre essas vantagens, destaca-se o uso do HTTP para transporte das informações. Com ele tem-se a segurança na transmissão de dados e ainda evita-se problemas com *firewalls*.

### **4.4 Considerações Finais**

Concluindo, percebe-se a facilidade de trabalhar com esta arquitetura utilizando o SOAP. Um dos motivos que mais impulsiona a disseminação desta tecnologia é a simplicidade notada para implementar aplicações distribuídas.

Optou-se pela ferramenta Delphi, por ser uma ferramenta RAD com diversos recursos e protocolos existentes para o desenvolvimento na arquitetura Multicamadas, entre estes, o protocolo SOAP.

## 5 UML

Devido à rápida percepção dos benefícios da OO (Orientação a Objetos), verificou-se o surgimento de diferentes propostas de análise e projeto orientados a objeto, tanto nos meios acadêmicos como nos empresariais. As primeiras propostas de notações para a Análise OO surgiram no início da década de 90 e logo após houve uma miscelânea de notações, gerando um excesso de propostas para a diagramação e construção de sistemas com técnicas OO (BOOCH, 2000. p. XVI-XX).

Entre as propostas de Análise OO mais robustas, destacam-se os seguintes autores:

- Grady Booch - com o método Booch
- James Rumbaugh - com o método OMT (Object Modeling Technique)
- Ivar Jacobson – com OOSE (Object-Oriented Software Engineering)
- Peter Coad / Edward Yourdon - com Análise Baseada em Objetos

Para que fosse definido e adotado um padrão universal, o consórcio OMG<sup>14</sup> (*Object Management Group*) aprovou a UML (*Unified Modeling Language*) em 1997. A UML é uma mistura de sintaxes gráficas pré-existentes, com alguns elementos removidos e outros elementos adicionados com o objetivo de torná-la mais expressiva. A UML é destinada para visualizar, especificar, construir e documentar sistemas complexos (BEZERRA, 2002).

A UML é adequada para a modelagem de sistemas, cuja abrangência poderá incluir desde sistemas de informação corporativos a serem distribuídos em aplicações baseadas em Web, até sistemas complexos embutidos de tempo real (BOOCH, 2000).

---

14 Consórcio internacional de empresas que define e ratifica padrões na área da orientação a objetos. (www.omg.org) Disponível em < <http://www.omg.org> >. Acesso em 12 ago. 2004.

Conforme Jones (2001) existem sete pontos importantes da UML que foram definidos pelos próprios autores:

1. Prover aos usuários uma linguagem de modelagem visual, a fim de desenvolver e compartilhar modelos significativos;
2. Prover mecanismos de extensibilidade e especialização para ampliar os conceitos centrais;
3. Ser independente de linguagens de programação e processos de desenvolvimento particulares;
4. Prover uma base formal para entendimento da linguagem de modelagem;
5. Estimular o crescimento do mercado de ferramentas orientadas a objetos;
6. Suportar conceitos de desenvolvimento de nível mais alto, tais como colaborações, estruturas, modelos e componentes;
7. Integrar as melhores práticas.

A UML, segundo Furlan (1998), pode ser usada ao longo do ciclo de vida de desenvolvimento de um sistema para:

1. Mostrar as fronteiras de um sistema e suas funções principais utilizando atores e casos de uso;
2. Ilustrar a realização de casos de uso com diagramas de interação;
3. Representar uma estrutura estática de um sistema utilizando diagramas de classe;
4. Modelar o comportamento de objetos com diagramas de transição de estado;
5. Revelar a arquitetura de implementação física com diagramas de componente e de implantação;
6. Estender sua funcionalidade através de estereótipos.

## **5.1 Evolução da UML**

Os métodos Booch e OMT estavam crescendo independentemente e sendo reconhecidos pela comunidade usuária como métodos de classe mundial. Seus autores, respectivamente, Grady Booch e James Rumbaugh juntaram forças através da Rational Corporation para forjar uma unificação completa de seus trabalhos. Em outubro de 1995, lançaram um rascunho do Método Unificado na versão 0.8, sendo esse o primeiro resultado concreto de seus esforços.

No ano de 1996, Ivar Jacobson também se integrou ao grupo da Rational Corporation para complementar a UML, com seu método OOSE (figura 5.1) (FURLAN, 1998. p. 31-32).

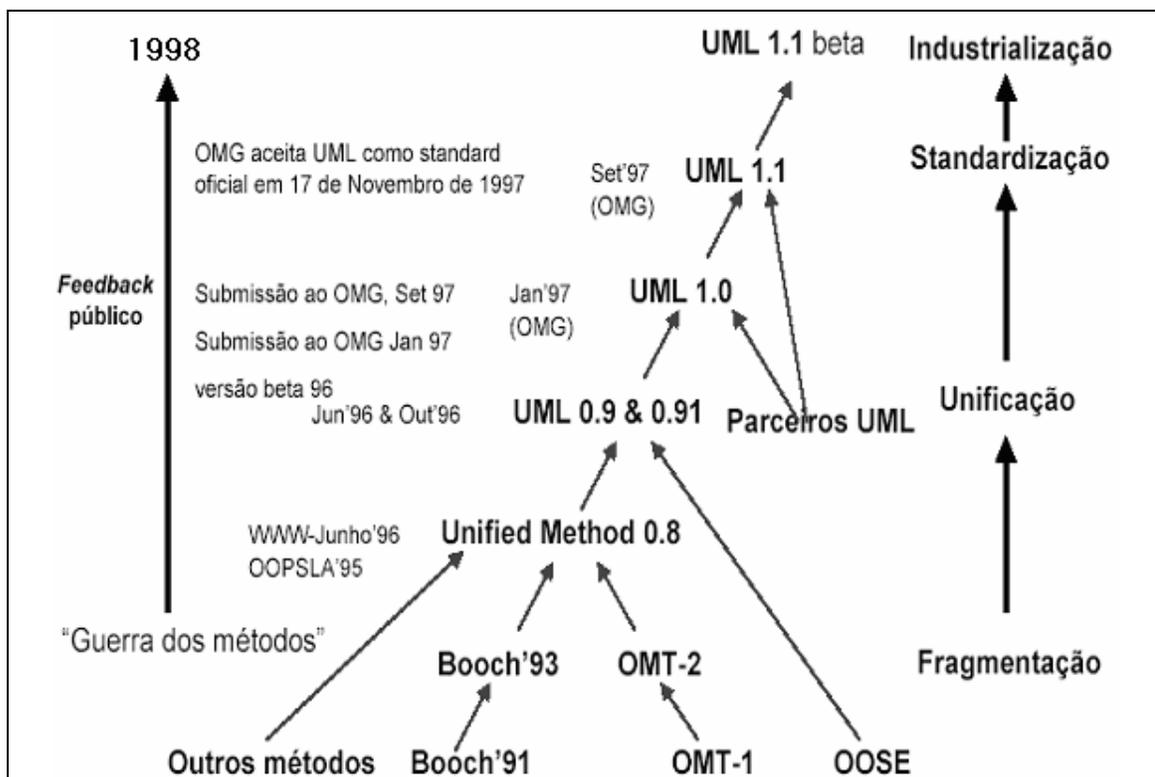


Figura 5.1 – Evolução da UML

Atualmente, o consórcio OMG está fazendo upgrade da UML para versão 2.0. Esta versão foi oficializada em 2003, mas a versão final deverá ser liberada somente no final de 2004.<sup>15</sup> (figura 5.2).

<sup>15</sup> Disponível em < <http://www.uml.org> >. Acesso em 20 set. 2004.

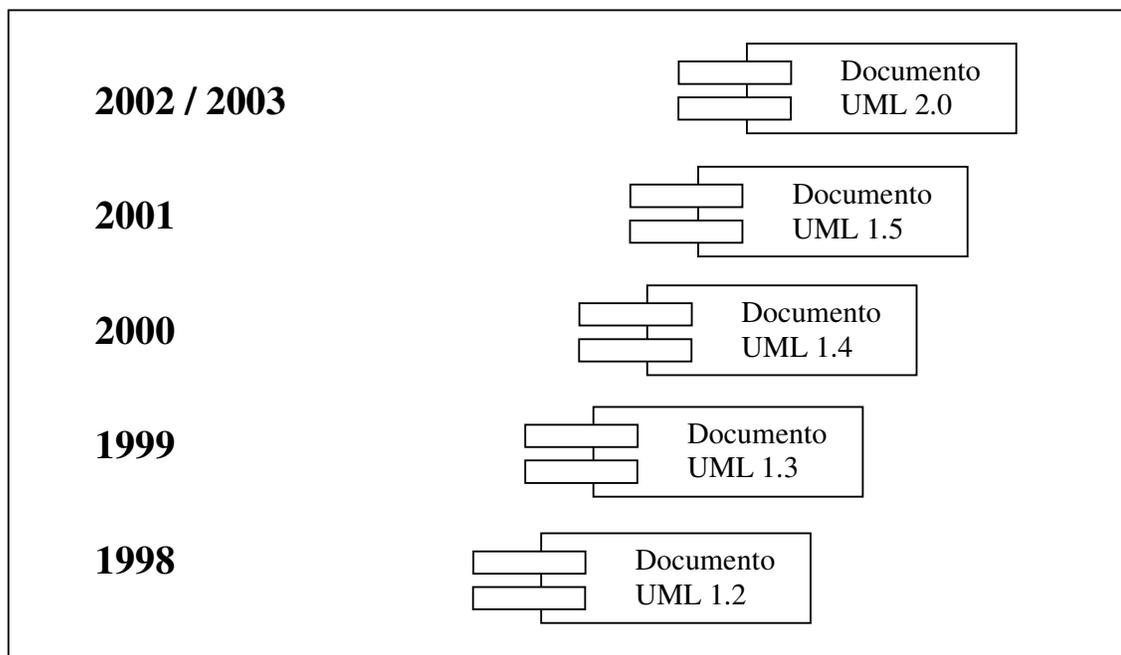


Figura 5.2 – A Evolução mais Recente

## 5.2 Diagramas da UML

Os artefatos gráficos produzidos durante o desenvolvimento de um sistema de software são definidos através da utilização dos diagramas da UML. Diagrama é a representação gráfica de um conjunto de elementos, geralmente representados como gráficos de itens e relacionamentos. É através dos diagramas que se pode visualizar o sistema (BEZERRA, 2002).

É difícil um único tipo de diagrama representar o ciclo de vida de um projeto, por isso, a UML disponibilizou vários tipos de diagramas para mostrar a estrutura e comportamento de um sistema sob várias perspectivas.

A UML é composta por nove diagramas e estes subdivididos em: (i) **diagramas estruturais**: são usados para visualizar, especificar, construir e documentar os aspectos **estáticos** de um sistema e os (ii) **diagramas comportamentais**: são utilizados para visualizar, especificar, construir e documentar os aspectos **dinâmicos** de um sistema (BOOCH, 2000 p. 93-97).

Os diagramas também podem ser subdivididos entre as fases de um projeto: (i) Análise – objetiva entender o problema e fazer um levantamento e/ou uma especificação dos requisitos como preparação para o projeto; (ii) Projeto – é o processo de refinamento e acréscimo de detalhes que faltaram na análise, e quando determina-se “como” o sistema funcionará para atender os requisitos; (iii) Implementação – é uma atividade na qual o sistema é codificado e/ou traduzido à linguagem de programação e a arquitetura selecionada para o sistema (BEZERRA, 2002).

### 5.2.1 Análise

É na fase da análise que tem-se o objetivo de entender o problema e fazer um levantamento e/ou uma especificação dos requisitos como uma preparação para o projeto.

#### A) Diagrama de Caso de Uso (*Use Case*)

O diagrama de caso de uso mostra um conjunto de casos de uso, de atores e seus relacionamentos. Este diagrama é usado para a organização e a modelagem do comportamento do sistema, com o qual descreve-se a visão externa do sistema e as suas interações com o mundo exterior, representando uma visão de alto nível da funcionalidade **comportamental** mediante o recebimento de requisições dos atores externos (usuário, outro sistema, um hardware, etc.) (figura 5.3) (BOOCH, 2000. p. 95). A modelagem do *use case* é utilizada não somente para capturar os requisitos de um novo sistema, mas também para desenvolver novas versões de um sistema.

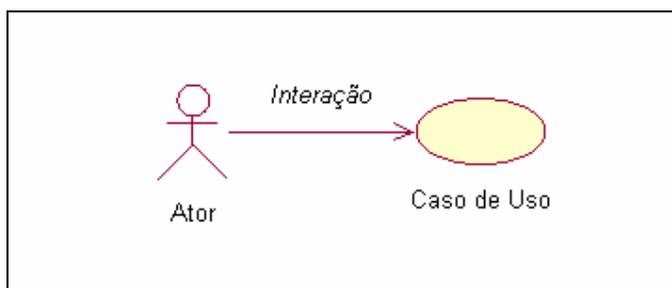


Figura 5.3 – Diagrama de Caso de Uso

#### B) Diagrama de Classes

O diagrama de classes é composto de classes, interfaces, colaboração e relacionamentos (dependência, generalização e associação) e usado para ilustrar a visão estática do projeto de um sistema, sendo um diagrama **estrutural**.

Os diagramas de classes são os diagramas mais encontrados na modelagem de sistemas orientados a objeto e um diagrama de objetos representa a instância de um diagrama de classe (figura 5.4) (BOOCH, 2000. p. 94).

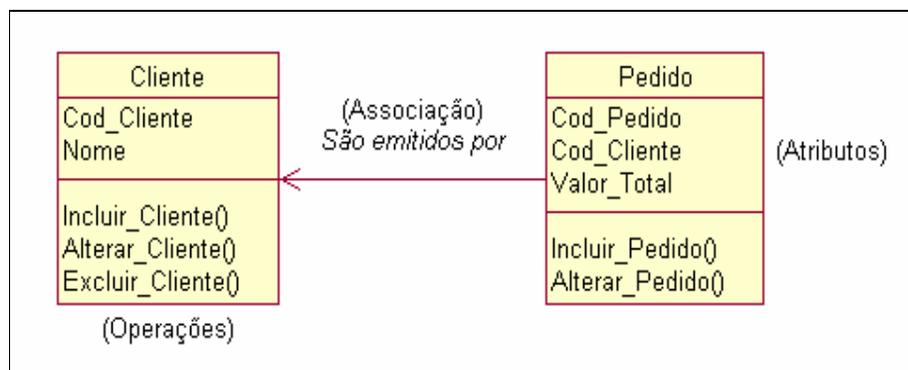


Figura 5.4 – Diagrama de Classes

### C) Diagrama de Objetos

O diagrama de objetos é um diagrama **estrutural** que mostra um conjunto de objetos e seus relacionamentos. É usado para ilustrar a estrutura dos dados.

### 5.2.2 Projeto

Na fase do projeto é feito o processo de refinamento e acréscimo de detalhes que faltaram na análise.

#### A) Diagrama de Seqüência

O diagrama de seqüência é um diagrama **comportamental** que mostra uma interação, dando ênfase à ordenação temporal das mensagens. Um diagrama de seqüência mostra um conjunto de objetos e as mensagens enviadas e recebidas por esses objetos (figura 5.5) (BOOCH, 2000. p. 96).

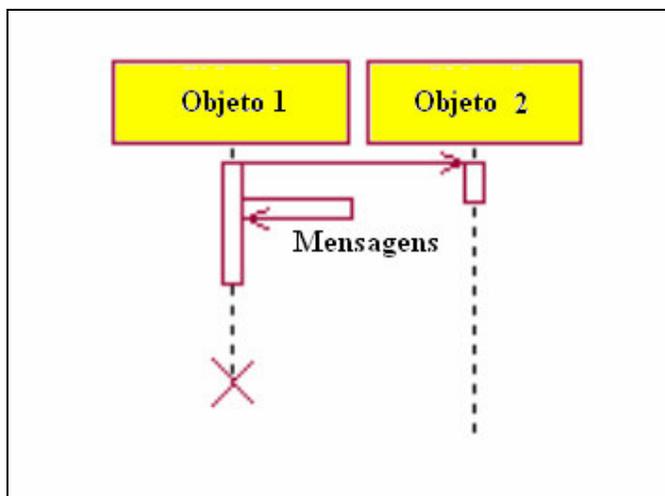


Figura 5.5 – Diagrama de Seqüência

### B) Diagrama de Colaboração

O diagrama de colaboração também é um diagrama **comportamental** que mostra uma interação, dando ênfase à organização estrutural de objetos que enviam e recebem mensagens. Este diagrama ilustra um conjunto de objetos, as conexões existentes entre os objetos e as mensagens enviadas e recebidas pelos objetos.

### C) Diagrama de Estados

O diagrama de estados é um diagrama **comportamental** que mostra uma máquina de estados, que consiste de estados, transições, atividades e dando ênfase ao comportamento ordenado por eventos de um objeto. Este diagrama refere-se ao diagrama de transição de estado da análise estruturada para o projeto orientado a objeto (JONES, 2001).

### D) Diagrama de Atividades

O diagrama de atividades também é um diagrama **comportamental** que mostra o fluxo de uma atividade para outra em um sistema. Uma atividade mostra um conjunto de atividades, o fluxo seqüencial ou ramificado de um atividade para outra e os objetos que realizam ou sofrem ações.

### 5.2.3 Implementação

Na fase de implementação o sistema é codificado e/ou traduzido à linguagem de programação.

#### A) Diagrama de Componentes

O diagrama de componentes é um diagrama **estrutural** que mostra um conjunto de componentes e seus relacionamentos. Estes diagramas estão relacionados aos diagramas de classes, pois tipicamente um componente mapeia uma ou mais classes, interfaces ou colaborações.

#### B) Diagrama de Implantação (*Deployment*)

O diagrama de implantação é outro diagrama **estrutural** que mostra um conjunto de nós e seus relacionamentos. Estes estão relacionados aos diagramas de componentes, pois tipicamente um nó contém um ou mais componentes.

## 5.3 As Visões de um Sistema

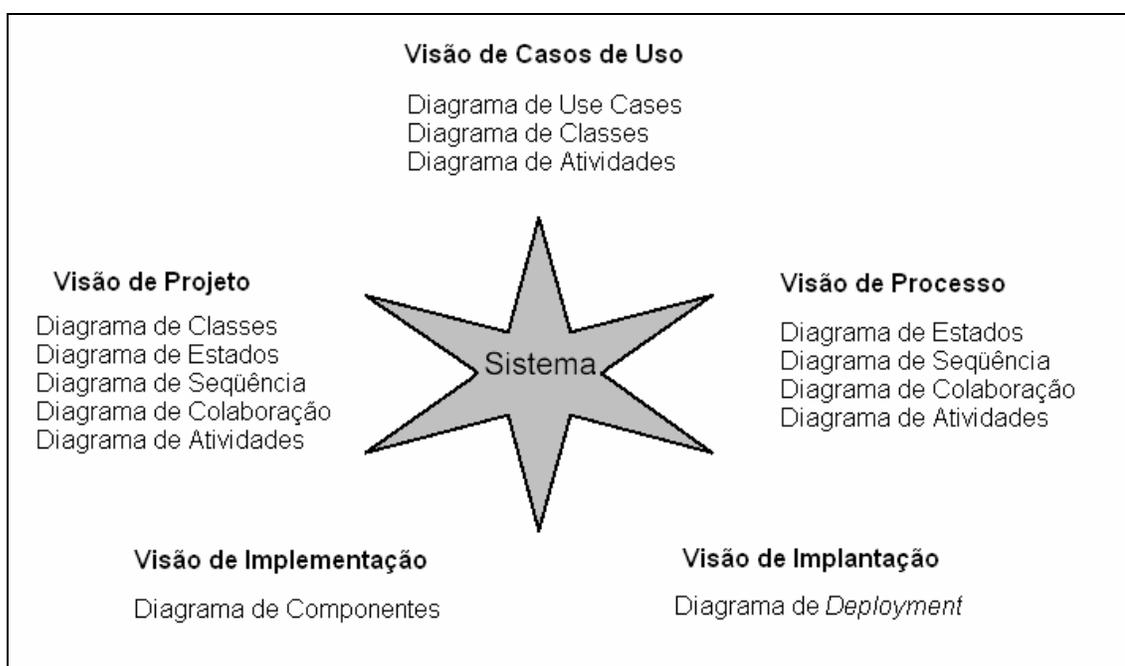
O desenvolvimento de um sistema complexo demanda que seus desenvolvedores tenham a possibilidade de examinar e estudar esse sistema a partir de diversas perspectivas. Os autores da UML sugerem que um sistema pode ser escrito por cinco visões interdependentes. Conforme Bezerra (2002) as visões propostas são as seguintes:

- **Visão de Casos de Usos:** descreve o sistema de um ponto de vista externo como um conjunto de interações entre o sistema e os agentes externos ao sistema. Esta visão é criada inicialmente e direciona o desenvolvimento das outras visões do sistema;
- **Visão de Projeto:** enfatiza as características do sistema que dão suporte, tanto estrutural quanto comportamental, às funcionalidades externamente visíveis do sistema;
- **Visão de Processo:** esta visão enfatiza as características de concorrência (paralelismo), sincronização e desempenho do sistema;
- **Visão de Implementação:** abrange o gerenciamento de versões do sistema, construídas através do agrupamento de módulos (componentes) e subsistemas;

- **Visão de Implantação:** corresponde à distribuição física do sistema em seus subsistemas e a conexão entre essas partes.

Dependendo das características e da complexidade do sistema, nem todas as visões precisam ser construídas. Por exemplo, se o sistema tiver de ser instalado em um ambiente computacional de processador único, não há necessidade da visão de implantação.

Na figura 5.6, tem-se os diagramas da UML subdivididos nas cinco visões, podendo certos diagramas estar repetidos nas visões.



**Figura 5.6 – Visões de um Sistema**

## 6 ESTUDO DE CASO

O objetivo principal deste trabalho é aplicar a tecnologia multicamadas na implementação de um sistema de ERP. Com isto, espera-se obter as vantagens oriundas da arquitetura multicamadas, tais como a segurança, a escalabilidade, a manutenibilidade, a organização arquitetural, o desempenho e a confiabilidade do sistema.

O estudo de caso consiste da modelagem e implementação dos módulos de Faturamento e Financeiro de um sistema de ERP, utilizando na prática a arquitetura multicamadas e as outras tecnologias e linguagens analisadas nos capítulos anteriores, tais como os componentes SOAP e *Web Services* do Delphi e a linguagem de modelagem unificada (UML).

A escolha do protocolo SOAP foi motivada pela melhor relação simplicidade/firewall, se comparado aos outros protocolos de comunicação. Aliado a isto, destaca-se como ponto fundamental para a disseminação do SOAP a simplicidade de uso da tecnologia para a implementação de aplicações distribuídas. Por fim, usando este protocolo em conjunto com o *Web Services* do Delphi, tem-se ainda mais recursos.

Em síntese, a escolha pelo Delphi usando os componentes SOAP e *Web Services*, para o desenvolvimento de dois módulos do Sistema de ERP, foi determinada pela facilidade de aprendizagem, implementação e pelos recursos oferecidos.

Neste estudo de caso, conforme já mencionado serão implementados dois módulos de um sistema ERP. Como um sistema de ERP possui outros módulos, estes serão modelados e implementados futuramente em uma empresa, na qual a equipe de projetistas e desenvolvedores estão ambientados com o Delphi. Mudar para outra ferramenta de desenvolvimento envolveria custos elevados, como por exemplo: aquisição de novas licenças, treinamento da equipe técnica, entre outros. Sendo assim, este é mais um motivo para escolha do Delphi.

Para a modelagem do sistema ERP, será utilizada a UML, pois através desta, tem-se uma maior facilidade de representar e gerar a documentação do software e que abrange, através de seus diagramas, qualquer característica de um sistema. A UML também é aplicada em diferentes fases do desenvolvimento de um projeto, desde a especificação da análise de requisitos até a fase de implantação.

## 6.1 Metodologia de Desenvolvimento

A metodologia de desenvolvimento usada será a de “**Ciclo de Vida em Espiral**”, na qual o processo de desenvolvimento é representado como uma espiral, ao invés de uma seqüência de atividades. Este modelo define quatro quadrantes, nos quais as atividades (gerenciais ou técnicas) de um projeto são executadas durante um ciclo na espira, conforme figura 6.1 (VASCONCELOS, 2002):

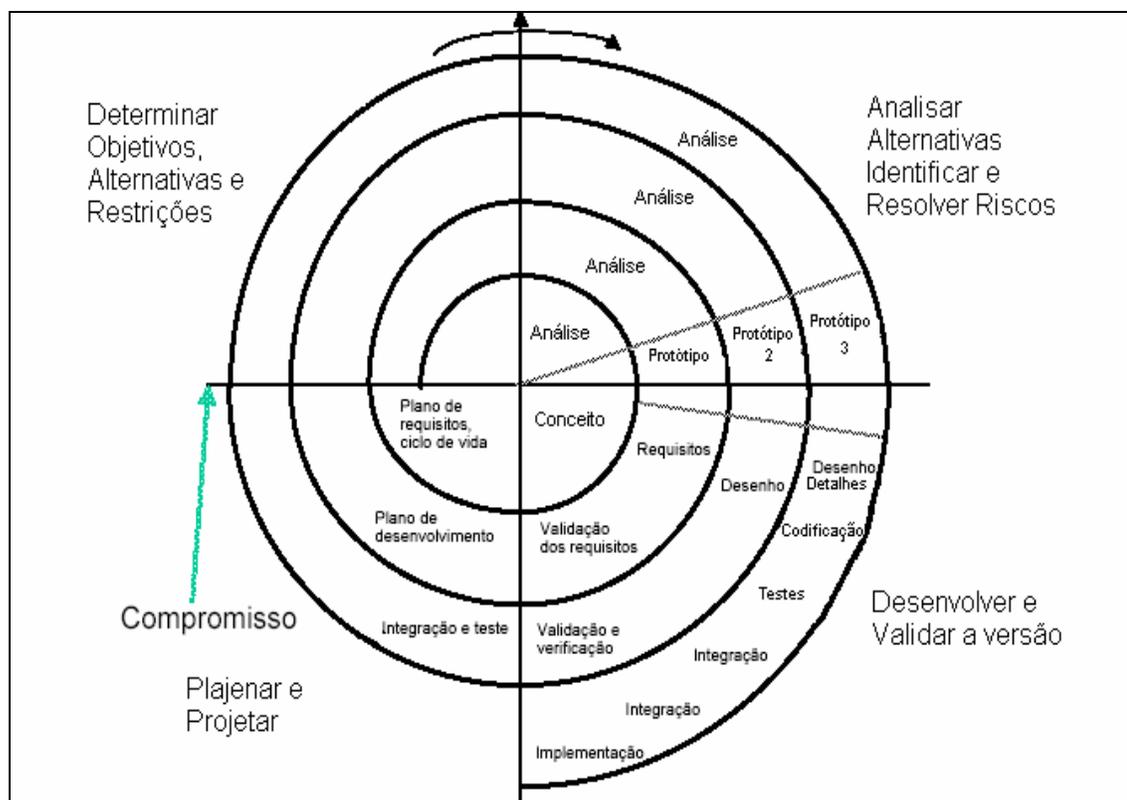


Figura 6.1 – Metodologia Espiral

- **Determinar objetivos, alternativas e restrições:** objetivos específicos para cada etapa são identificados. Alternativas para realizar os objetivos e restrições são encontradas;

- **Analisar alternativas e identificar e/ou resolver riscos:** os principais riscos são identificados, analisados e buscam-se meios para reduzir esses riscos;
- **Desenvolver e validar a versão corrente do produto:** um modelo apropriado para o desenvolvimento é escolhido, o qual pode ser qualquer um dos modelos de ciclo de vida;
- **Planejar e projetar:** o projeto é revisto e o próximo ciclo da espiral é planejado.

## 6.2 Requisitos do Software

Uma compreensão completa dos requisitos de software é de fundamental importância para que um sistema seja bem-sucedido como um todo. Um sistema bem projetado e codificado, mas que tenha uma análise das necessidades mal elaborada ou incompleta, acaba prejudicando o usuário e criando problemas futuros para a manutenção do mesmo.

A função da análise de requisitos é um processo de descoberta, refinamento, modelagem e especificação do sistema. O escopo do software, inicialmente estabelecido pelo engenheiro do sistema e refinado durante o planejamento do software, é aperfeiçoado e aprimorado em detalhes e este se tornará a base para todas as atividades que virão em seguida (PRESSMAN, 1995. p.231 e 232).

## 6.3 Módulos do Sistema ERP

O ERP é formado por um conjunto de módulos integrados, que facilitam o fluxo das informações entre todas as atividades da empresa evitando redundância de dados e oferecendo funcionalidades para os diversos processos de negócios das organizações (figura 6.2).



**Figura 6.2 – ERP – Informação e Conhecimento**

Neste estudo de caso, o primeiro módulo a ser implementado será o do Faturamento e o segundo o módulo do Financeiro. O sistema ainda terá todas classes necessárias para o bom e correto funcionamento do mesmo, tais como: Cadastro de Usuários, Clientes, Fonecedores, Representantes, Transportadoras, Naturezas de Operação, Produtos, Forma de Pagamento, entre outros cadastros auxiliares.

### **6.3.1 Cadastros Básicos**

O sistema deverá ter vários cadastros, alguns com dados já pré-cadastros, para que o sistema torne-se funcional e de fácil uso para o usuário. Estas classes devem ter as funções básicas e/ou padrões de um sistema informatizado, tais como: inclusão, alteração, exclusão, consulta e pesquisa dos dados já registrados, podendo ser esta pesquisa através da visualização dos dados no vídeo ou na forma de um relatório. Dentre estes diversos cadastros, destacam-se os seguintes:

- **Cadastro de Usuários** – usuário é a pessoa que vai estar utilizando o sistema de ERP. Nesta classe estarão os atributos tais como: Login e Senha (figura 6.3);

Usuarios
Usuario_ID Login Senha
Incluir_Usuarios() Alterar_Usuarios() Excluir_Usuarios()

**Figura 6.3 – Classe de Usuários**

- **Cadastro de Clientes** – cliente é a pessoa e/ou empresa que vai comprar o produto. Nesta classe estarão os atributos tais como: razão social, CNPJ, fone, endereço, etc. (figura 6.4);

Clientes
Cliente_ID Nome Fone Rua Bairro CEP Cidade_ID Tipo_Pessoa CNPJ_CPF Ins_Estadual RG Observacao
Incluir_Clientes() Alterar_Clientes() Excluir_Clientes()

**Figura 6.4 – Classe de Clientes**

- **Cadastro de Fornecedores** – fornecedor é a pessoa e/ou empreendimento da qual a empresa vai comprar os produtos e repô-los ao estoque. Nesta classe estarão os atributos tais como: razão social, CNPJ, fone, endereço, etc. (figura 6.5);

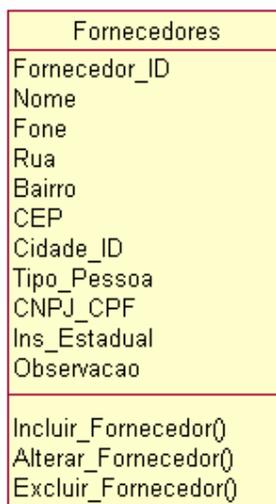


Figura 6.5 – Classe de Fornecedores

- **Cadastro de Representantes** – representante é a entidade que representa a empresa e/ou vende a mercadoria. Nesta classe constarão atributos tais como: nome, fone, endereço, percentual de comissão (figura 6.6);

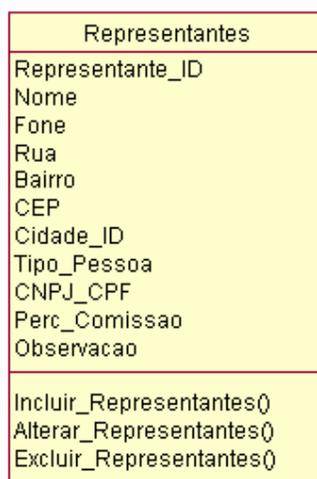


Figura 6.6 – Classe de Representantes

- **Cadastro de Transportadoras** – transportadora que irá transportar a mercadoria negociada com o cliente. Nesta classe estarão os atributos tais como: razão social, CNPJ, fone, endereço, etc. (figura 6.7);

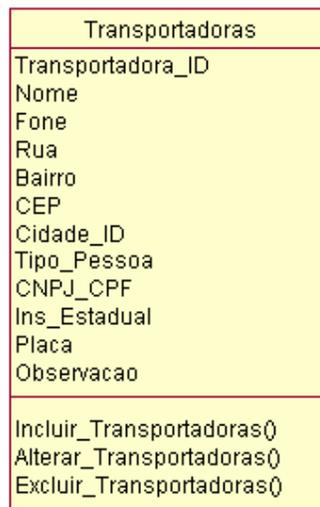


Figura 6.7 – Classe de Transportadoras

- **Cadastro das Naturezas de Operação** – natureza de operação que vai constar na nota fiscal, conforme a legislação. Indica que tipo de mercadoria está sendo negociada e como ela está sendo transferida de uma empresa para outra. Nesta classe estarão os atributos tais como: natureza de operação e a descrição da mesma (figura 6.8);

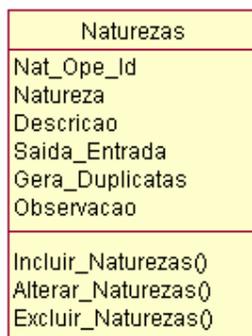


Figura 6.8 – Classe das Naturezas de Operação

- **Cadastro de Produtos** – neste cadastro estarão todos os produtos que a empresa irá vender e/ou negociar. Nesta classe estarão os atributos tais como: código do produto, descrição, preço de compra e venda, classificação fiscal, situação tributária, etc. (figura 6.9);

Produtos
Item_ID
Descricao
Fornecedor_ID
Preco_Compra
Preco_Venda
Peso
Unidade
Classif_Fiscal
Observacao
Incluir_Produtos()
Alterar_Produtos()
Excluir_Produtos()

**Figura 6.9 – Classe dos Produtos**

- **Cadastro das Formas de Pagamento** – nesta classe estarão todas as formas de pagamento. Formas de pagamento como: “a vista” e as formas a prazo do tipo “30/60 dias”, “10/20/30 dias”, etc. (figura 6.10);

Formas_Pagamento
Form_Pgto_ID
Descricao
Complemento
Percentual
Incluir_Formas_Pagamentos()
Alterar_Formas_Pagamentos()
Excluir_Formas_Pagamento()

**Figura 6.10 – Classe das Formas de Pagamento**

- **Cadastro das Cidades** – as cidades usadas (referenciadas) nas classes: clientes, fornecedores, representantes, etc, estarão na classe cidades (figura 6.11);

Cidades
Cidade_ID Descricao Estado_ID
Incluir_Cidades() Alterar_Cidades() Excluir_Cidades()

**Figura 6.11 – Classe das Cidades**

- **Cadastro dos Estados** – os estados usados (referenciados) na classe de cidades estarão na classe estados (figura 6.12);

Estados
Estado_ID Descricao Pais_ID
Incluir_Estados() Alterar_Estados() Excluir_Estados()

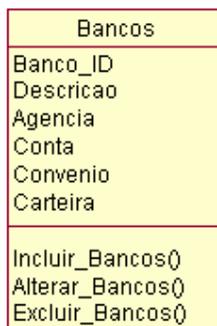
**Figura 6.12 – Classe dos Estados**

- **Cadastro de Países** – os países usados (referenciados) na classe de estados estarão na classe países (figura 6.13);

Países
Pais_ID Descricao
Incluir_Paises() Alterar_Paises() Excluir_Paises()

**Figura 6.13 – Classe dos Países**

- **Cadastro de Bancos** – Nesta classe estarão os bancos, que vão emitir os bloquitos da cobrança de clientes (figura 6.14);



**Figura 6.14 – Classe dos Bancos**

### 6.3.2 Faturamento

A função principal do módulo do faturamento é a de controlar todas as entradas e saídas de uma empresa, ou seja, todo produto deve ter uma nota (documento fiscal) para comprovar sua origem/destino e conseqüentemente o recolhimento dos impostos. Documentos fiscais podem ser: (i) Nota Fiscal de Saída, a qual é o documento que comprova que um produto saiu do estabelecimento; (ii) Nota Fiscal de Entrada, a qual é o documento que comprova que um produto entrou no estabelecimento; entre outros documentos. O capítulo sete irá apresentar os requisitos e a modelagem do módulo de Faturamento.

### 6.3.3 Financeiro

A função principal do Financeiro é a de controlar o fluxo da cobrança de títulos emitidos pelo estabelecimento ou recebidos de outro estabelecimento através da venda e/ou compra de um produto via nota fiscal. O capítulo oito irá apresentar os requisitos e a modelagem do módulo Financeiro.

## 7 FATURAMENTO

A função principal do módulo do Faturamento é a de controlar todas as saídas de uma empresa, ou seja, todo produto (matéria-prima, produto acabado, patrimônio) deve ter uma nota (documento fiscal impresso) para comprovar sua origem/destino.

Os documentos fiscais podem ser: (i) Nota Fiscal de Saída, que é o documento que comprova que um produto saiu do estabelecimento; (ii) Nota Fiscal de Entrada, que é o documento que comprova que um produto entrou no estabelecimento. Através destas notas, é feita a comprovação e o recolhimento dos impostos, a fim de evitar multas e constrangimentos com a fiscalização de tributos.

As notas fiscais podem representar a compra de um produto, a venda de um produto, a transferência de um produto de um estabelecimento para o outro ou a prestação de um serviço. Além da função principal do faturamento, o sistema ainda deve atender os seguintes requisitos:

- **Integração com o Financeiro** – os documentos emitidos podem gerar ou não um título (comprovante de uma transação realizada entre estabelecimentos), através da emissão de uma nota fiscal de venda ou de compra. Os títulos são emitidos com base nas formas de pagamento, podendo ser estas formas como: (i) a vista ou apresentação – pagamento realizado no ato; (ii) a prazo – pagamento a ser realizado conforme os dias combinados entre vendedor e cliente;
- **Cálculo da Nota** – o sistema deve calcular os valores, os totais da nota fiscal e os impostos conforme as políticas fiscais vigentes. É calculado o ICMS (Imposto sobre Circulação de Mercadorias e Serviços), o IPI (Imposto sobre Produtos Industrializados), frete, seguro, etc;
- **Controle de Comissões de Representantes** – no momento de emitir uma nota fiscal de venda, o sistema deve ter um processo de controle da comissão dos representantes, conforme o percentual combinado entre empresa e representante;

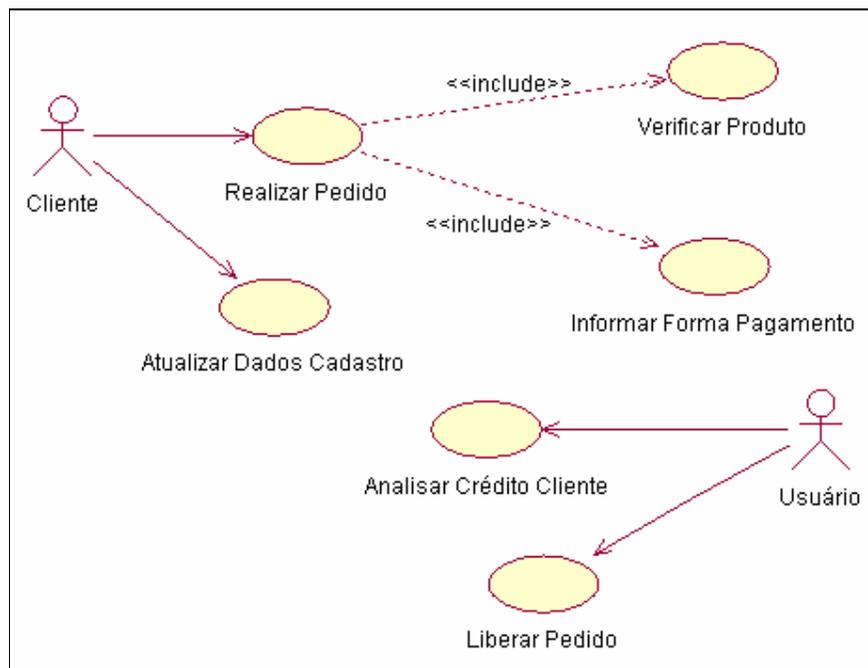
- **Impressões** – o software deve imprimir a nota fiscal com os dados conforme a legislação fiscal vigente e esta deve acompanhar a mercadoria até o destino, sendo o comprovante da compra e usada para fins da fiscalização tributária;
- **Relatórios** – o sistema deve fazer a impressão dos seguintes relatórios: (i) Total do Faturamento por dia e/ou mês – este relatório deve imprimir todas as vendas realizadas em um período; (ii) Relatório de Comissões – este irá imprimir os valores das comissões que devem ser pagas aos representantes; (iii) Vendas por Produto – este relatório deve imprimir as vendas por produto em um período; (iv) Cadastro de Clientes – este relatório irá imprimir a relação dos clientes com dados principais de cada um; (v) Cadastro de Fornecedores – este relatório irá imprimir a relação dos fornecedores com os dados principais de cada um. Antes da impressão de cada relatório, deverá ter uma tela de seleção a qual irá filtrar os registros, e nesta deve ter informações do tipo: nome do cliente ou fornecedor, data de emissão dos documentos, entre outras opções com base no relatório.

O restante do capítulo detalha como funciona o Controle dos Pedidos e a Emissão da Nota Fiscal.

## 7.1 Controle de Pedido

Os pedidos são efetivados pelos clientes via telefone ou via Internet, para isto, os itens do pedido (produtos à comprar) devem ser informados. Estes produtos estão na classe de `produtos`, a qual está relacionada aos itens do pedido. Mas antes do sistema confirmar a solicitação do produto pelo cliente, o sistema irá consultar a disponibilidade do produto. Estando o produto disponível, este será anexado ao pedido, caso contrário o sistema irá retornar uma mensagem ao cliente de que o produto não está disponível. Após a confirmação dos itens do pedido, ainda deverá ser informado a forma de pagamento e caso for necessário a atualização do cadastro do cliente.

Depois da solicitação do pedido, este é encaminhado para o usuário do sistema, o qual pode ou não liberar o pedido, dependendo da análise do crédito do cliente. Os processos envolvidos na solicitação do pedido podem ser vistos no caso de uso do `Controle de Pedido` (figura 7.1).



**Figura 7.1 – Caso de Uso - Controle de Pedido**

No diagrama de classes mostrado na figura 7.2, percebe-se as diversas classes associadas ao controle do pedido, sendo que, as classes principais são `Pedidos` e `Pedidos_Itens`.

O pedido é realizado pelo cliente informando os dados na classe `Pedidos`. Um pedido é composto por itens (`Pedidos_Itens`) que o cliente vai solicitar. Cada um dos itens de um pedido está relacionado a um produto que existe na classe de `Produtos`. Entre as classes `Pedidos` e `Pedidos_Itens` tem-se uma associação de composição. A informação de condição de pagamento está relacionada a classe `Formas_Pagamento`. As classes `Cidades`, `Estados` e `Países` estão relacionadas aos clientes e fornecedores por existir uma associação através do atributo `Cidade_ID`.

O pedido realizado pelo cliente precisa ser aprovado, dependendo do crédito do cliente. Esta aprovação será realizada pelo usuário do sistema através da invocação do método `Liberar_Pedido()` que irá alterar o atributo `Status` da classe de `Pedidos`.

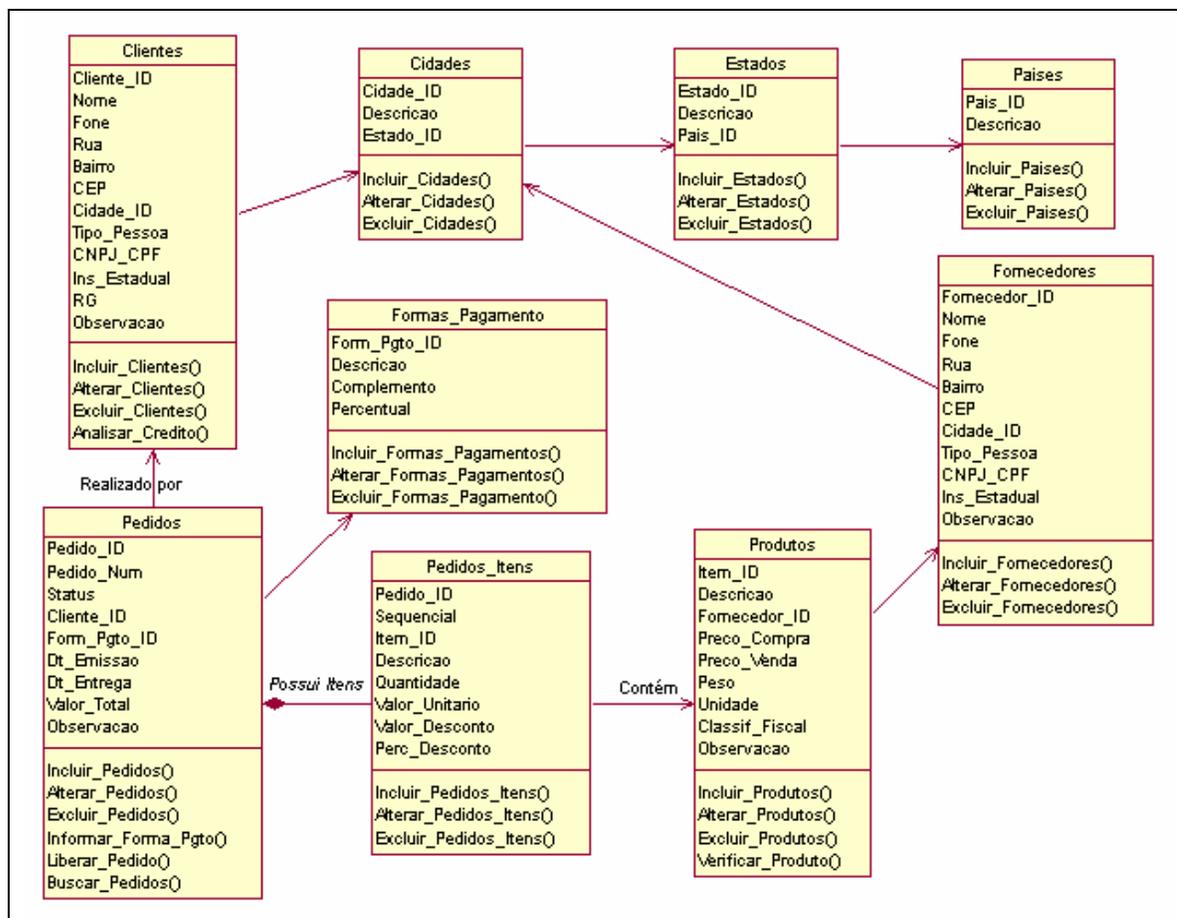


Figura 7.2 – Diagrama de Classes - Controle de Pedido

No diagrama de seqüência, figura 7.3, está demonstrada a seqüência lógica das operações envolvidas nos processos da realização do pedido. O cliente vai realizar o pedido informando os itens, ou melhor, vai solicitar os produtos disponíveis na tabela de `Produtos`. Após a inclusão dos itens o cliente deve informar a forma de pagamento, finalizando assim o processo da inclusão do pedido.

Após a inclusão do pedido, este está sujeito a aprovação através da análise do crédito do cliente, que será feita pelo usuário do sistema. Tendo o cliente seu crédito aprovado, o pedido é liberado através do fluxo `Liberar_Pedido` e os produtos solicitados serão separados e a entrega será feita após a emissão da nota fiscal .

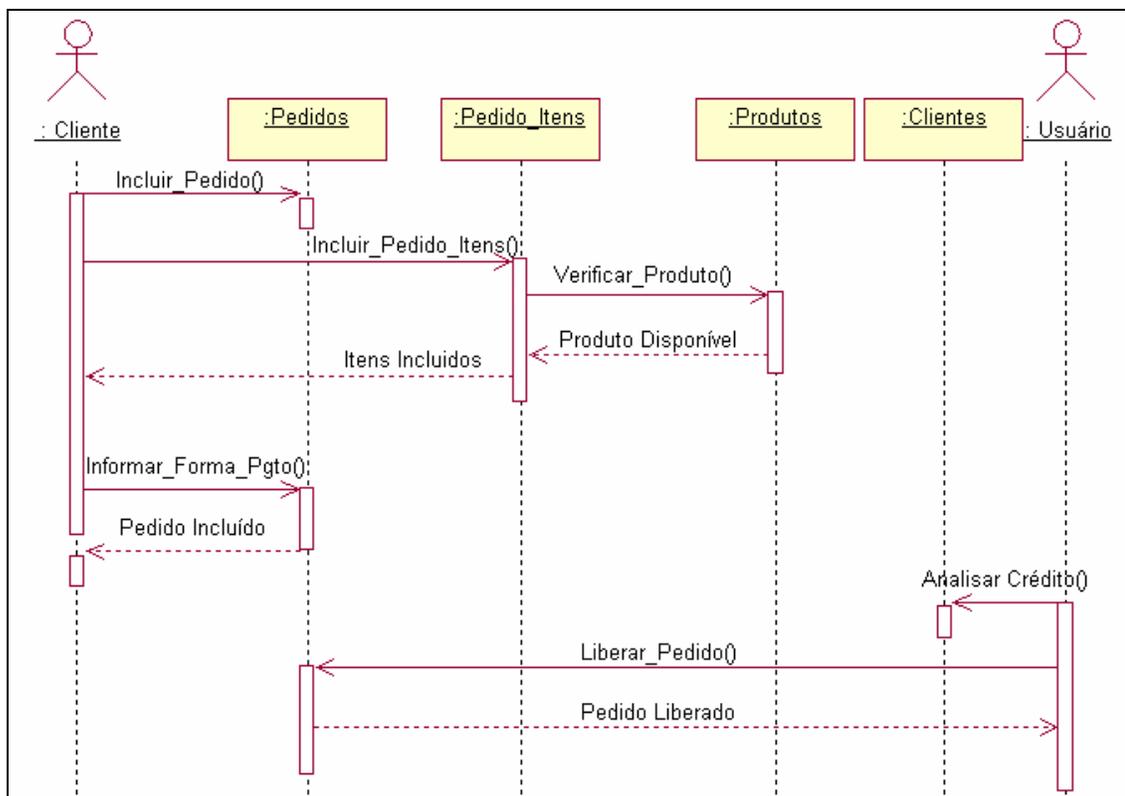


Figura 7.3 – Diagrama de Seqüência - Controle de Pedido

## 7.2 Emissão da Nota Fiscal

Após a liberação do pedido do cliente, a entrega do material solicitado será providenciada e o pedido será encaminhado para os trâmites da emissão da nota fiscal. Após a emissão da nota fiscal, as duplicatas também serão emitidas, as quais são integradas ao módulo do Financeiro. Estes processos podem ser vistos no caso de uso da Emissão da Nota Fiscal (figura 7.4).

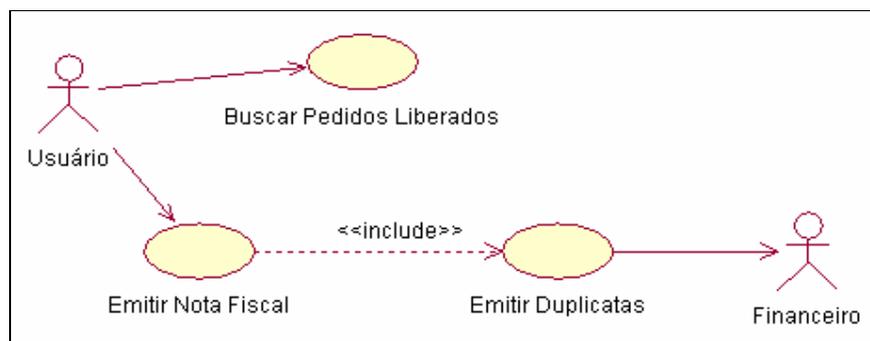


Figura 7.4 – Caso de Uso - Emissão da Nota Fiscal

Pelo diagrama de classes mostrado na figura 7.5 nota-se as associações entre as diferentes classes. Neste diagrama, as classes principais são `Notas` e `Notas_Itens` e todas as outras classes estão relacionadas em função destas.

O usuário do sistema consultará os pedidos liberados pelo processo anterior. Recebidos os pedidos o usuário emite a nota fiscal, representando pela associação “Emitido por” entre `Usuarios` e `Notas`. Ainda neste processo, deve ser confirmada a forma de pagamento (servirá para emissão das duplicatas e/ou bloqueto), deve ser informada a transportadora que vai transportar a mercadoria e ainda outros dados adicionais, tais como o representante, etc.

No processo da emissão da nota fiscal ainda deve ser informada a natureza de operação que indicará qual o tipo de nota fiscal está sendo emitida. No caso de ser uma nota proveniente de um pedido, a natureza de operação deve ser uma natureza de venda. Também existirão notas fiscais da compra de produtos, e por este motivo tem-se a classe de `fornecedores` relacionada a `notas`. Além destas, também podem existir notas fiscais de transferência de um produto para outro estabelecimento ou notas de serviço, entre outras.

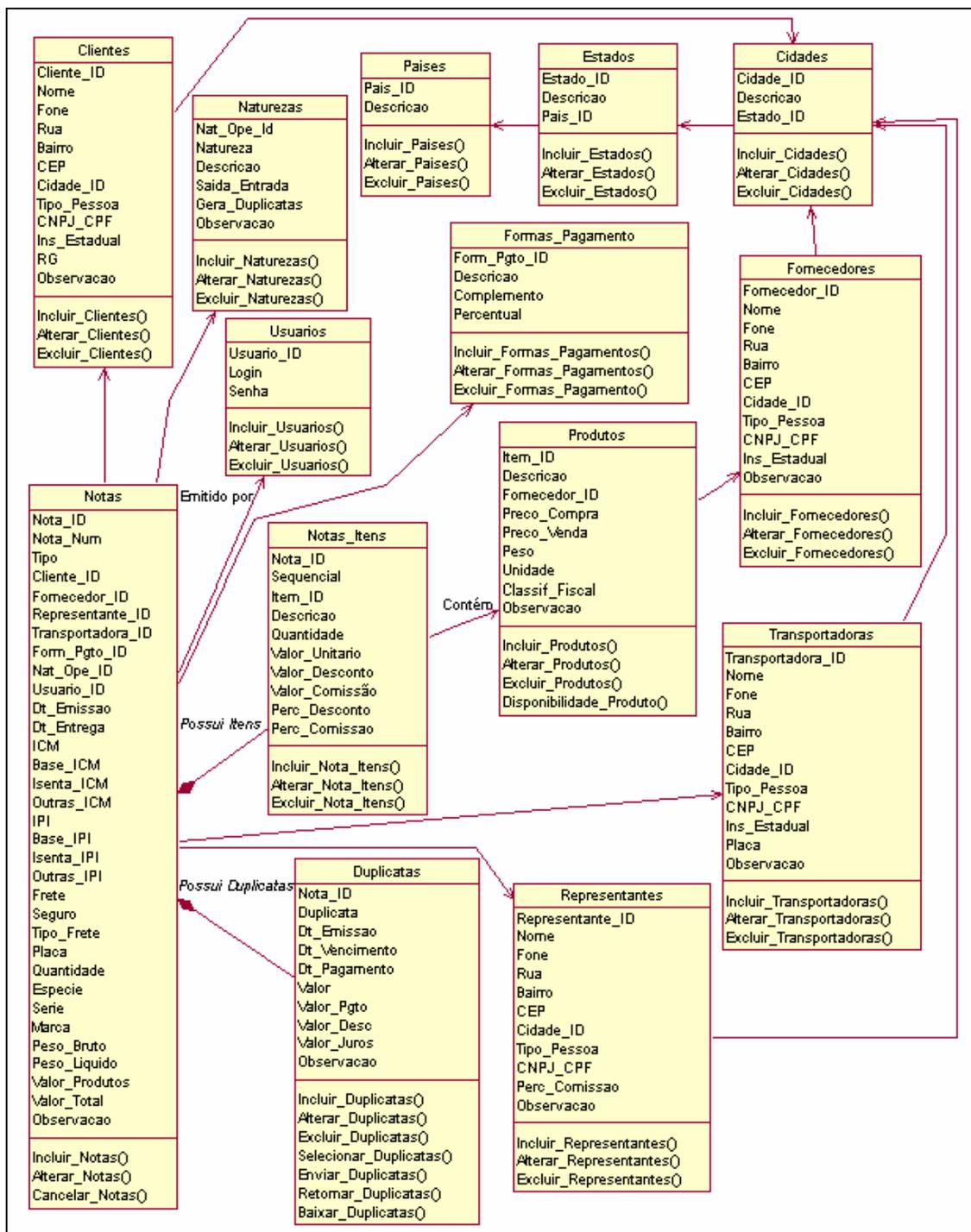


Figura 7.5 – Diagrama de Classes - Emissão da Nota Fiscal

No diagrama de seqüência da figura 7.6 está demonstrada a seqüência lógica das operações do processo da emissão da nota fiscal. O usuário irá verificar os pedidos liberados e para estes é emitida uma nota fiscal da venda de produtos, através do fluxo `Incluir_Notas`.

Junto ao processo da emissão da nota fiscal, tem-se também a geração das duplicatas, que são geradas de acordo com a forma de pagamento informada na nota fiscal e estas são encaminhadas automaticamente para o módulo do Financeiro. As formas de pagamento podem ser do tipo: (i) A vista - neste caso somente uma duplicata será gerada no valor da nota fiscal; (ii) 30/60/90 Dias – neste caso três duplicatas serão geradas, o valor total da nota será dividido por três e o vencimento da primeira duplicata será 30 dias após a emissão da nota e as outras 60 e 90 dias após a emissão. Podem existir outras formas de pagamento que igualmente vão estar na classe `Formas_Pagamento`.

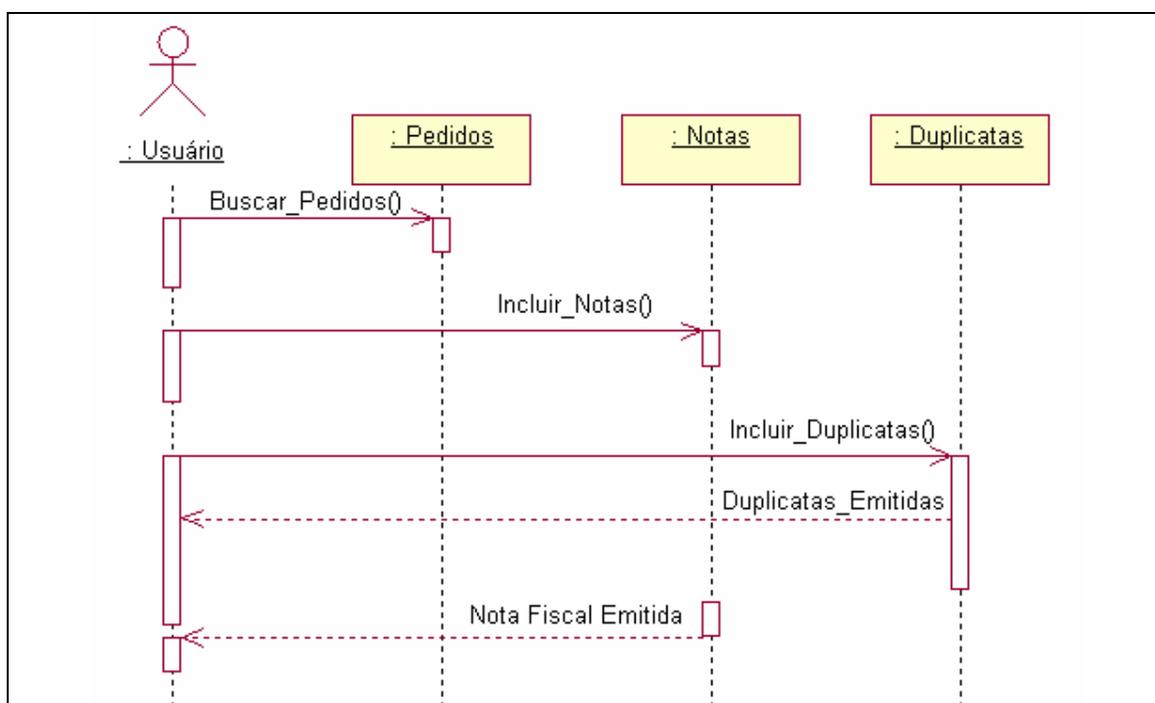


Figura 7.6 – Diagrama de Seqüência - Emissão da Nota Fiscal

## 8 FINANCEIRO

A função principal do módulo Financeiro é controlar a cobrança de títulos emitidos pelo estabelecimento ou recebidos de outro estabelecimento através da venda e/ou compra de um produto via nota fiscal.

Os títulos podem ser de diferentes formas: uma duplicata, um bloqueto, uma nota promissória, um cheque, etc, os quais são utilizados para realizar a cobrança de um cliente e/ou para fazer um pagamento à um fornecedor. Além da função principal do financeiro, o sistema ainda deve atender os seguintes requisitos:

- **Integração com o Faturamento** – os títulos emitidos através de uma nota fiscal vão estar disponíveis no financeiro (contas a pagar/receber) para realizar e/ou fazer a cobrança destes;
- **Lançamento e Emissão Manual** – o sistema deve ter a opção de lançar e emitir um título manualmente. Este título pode ser oriundo de uma cobrança ou de uma despesa realizada;
- **Cobrança Bancária** – o sistema deve ter a opção de remeter (enviar) os títulos abertos do “contas a receber” para uma instituição financeira, para agilizar o processo de cobrança. Assim que efetuado o pagamento pelo cliente, o banco envia uma remessa de retorno do pagamento e assim, este título será liquidado automaticamente pelo sistema. Este processo deve ser realizado pelo padrão do EDI (*Electronic Data Interchange* – troca eletrônica de informações);
- **Impressões** – o software deve imprimir a duplicata e/ou o bloqueto de cobrança com os dados da empresa emitente, do cliente, os valores e os demais dados necessários conforme a legislação tributária;
- **Relatórios** – o sistema deve fazer a impressão dos seguintes relatórios: (i) Relatório de Títulos do Contas a Receber – este irá imprimir os títulos abertos ou liquidados do contas a receber; (ii) Relatório de Títulos do Contas a Pagar – este

irá imprimir os títulos abertos ou liquidados do contas a pagar; (iii) Fluxo de Caixa – este relatório deve imprimir toda a movimentação de valores do contas a receber e do contas a pagar. Antes da impressão de cada relatório, deverá ter uma tela de seleção a qual irá filtrar os registros, e nesta deve ter informações do tipo: nome do cliente ou fornecedor, data de emissão dos documentos, entre outras opções com base no relatório.

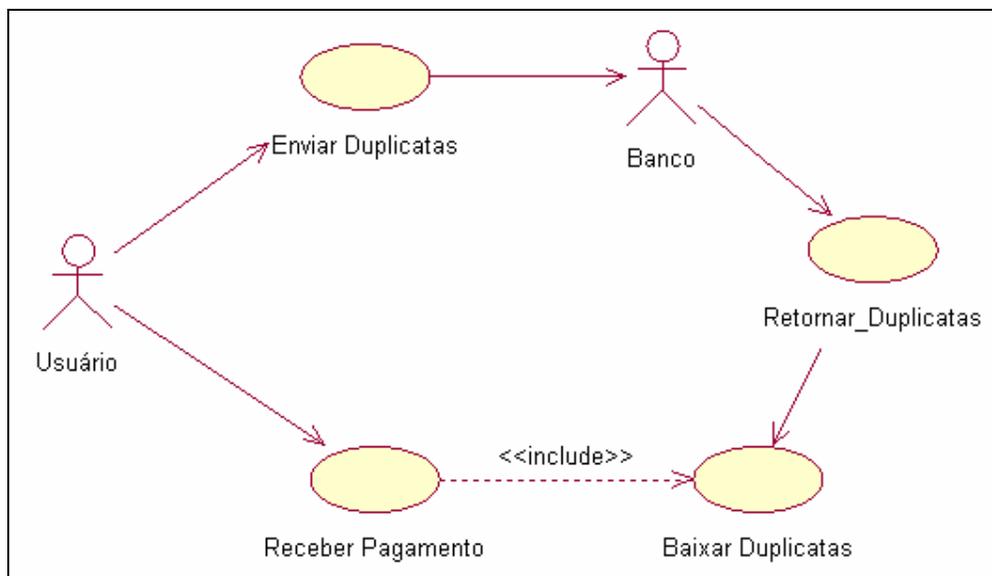
O módulo do Financeiro é subdividido em “Contas a Receber” e em “Contas a Pagar”, estes estão descritos nas seções a seguir.

## **8.1 Contas a Receber**

O módulo de Contas a Receber tem a função de controlar a cobrança dos títulos oriundos da venda de produtos realizado pelo estabelecimento. Além disto, também é responsável por efetuar qualquer outra cobrança de valores e fazer o registro destes.

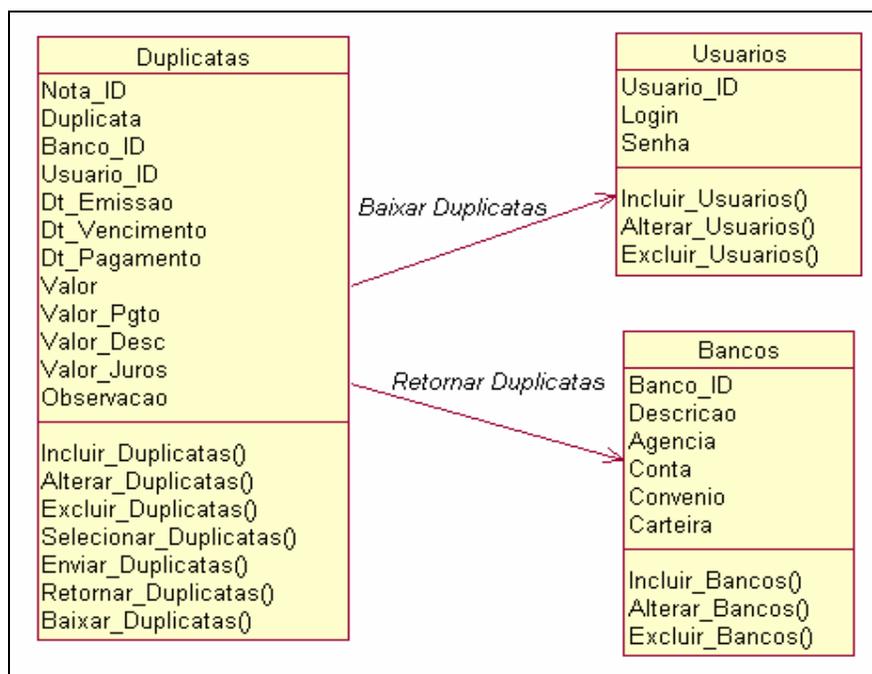
O pagamento pode ser realizado via bloqueto através da cobrança bancária EDI. Neste caso o usuário do sistema gera uma arquivo texto (TXT) no formato CNAB (Centro Nacional de Automação Bancária) e este arquivo é enviado ao banco, através de um software próprio da instituição bancária. O banco recebe o arquivo texto, emite o bloqueto de cobrança e o envia ao cliente, para que este efetue o pagamento. Após efetuado o pagamento pelo cliente, o banco retorna um arquivo texto para a empresa e o próprio sistema faz a baixa das duplicatas, facilitando assim o processo da cobrança e a baixa dos títulos (figura 8.1).

O usuário também pode receber o pagamento diretamente do cliente (no caso de não ser uma cobrança bancária). Após o recebimento do pagamento do cliente a baixa do título é realizada manualmente pelo usuário do sistema (figura 8.1).



**Figura 8.1 – Caso de Uso - Contas a Receber**

No diagrama de classes da figura 8.2 nota-se o relacionamento entre as classes *Duplicatas* e *Usuarios*. Neste o usuário realiza a baixa dos títulos recebidos através do cliente. O relacionamento entre *Duplicatas* e *Bancos* existe para demonstrar o vínculo entre a instituição financeira e as duplicatas enviadas.



**Figura 8.2 – Diagrama de Classes - Contas a Receber**

No diagrama de seqüência da figura 8.3 está demonstrada a seqüência lógica das operações do contas a receber. Neste diagrama percebe-se o fluxo `Selecionar_Duplicatas`, que fará a seleção das duplicatas que devem ser enviadas ao banco. Logo após a seleção, estas são enviadas ao banco pelo fluxo `Enviar_Duplicatas`. O banco as recebe e faz o devido processamento. Assim que o banco receber o pagamento do cliente, este faz o retorno das duplicatas a empresa, e estas são baixadas automaticamente pelo sistema.

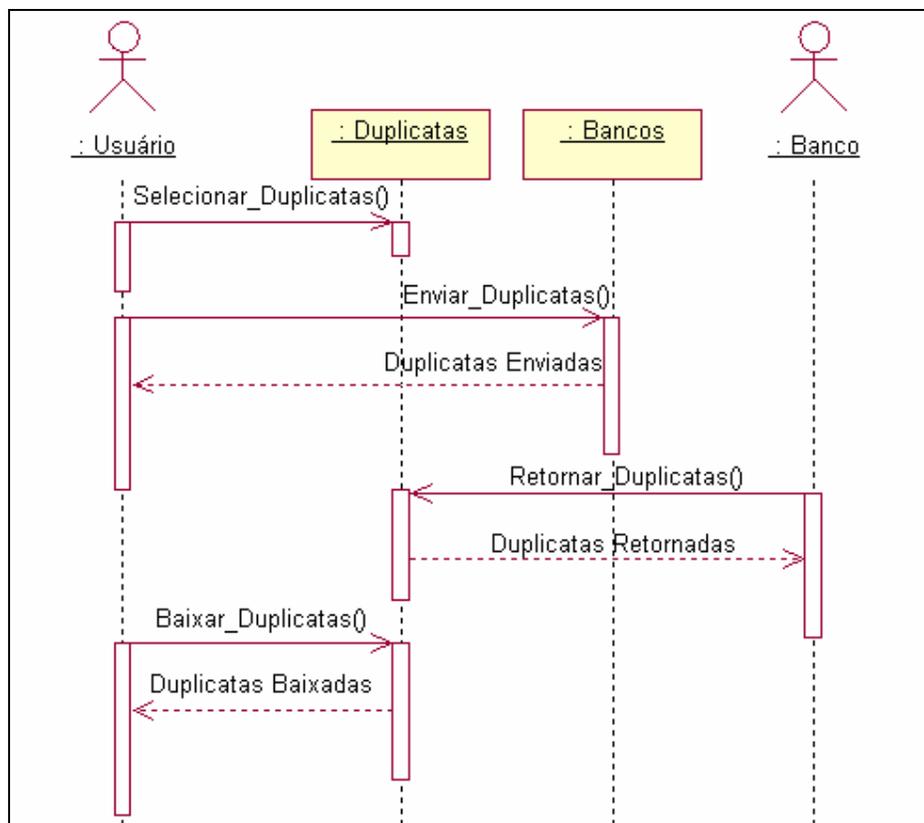


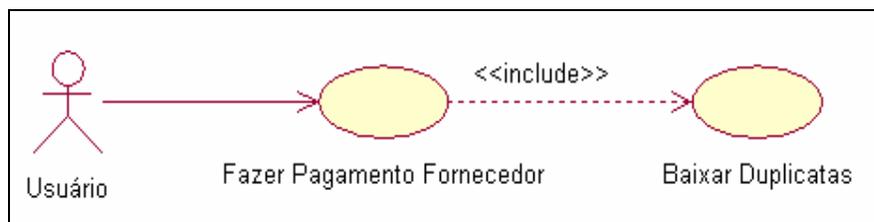
Figura 8.3 – Diagrama de Seqüência - Contas a Receber

## 8.2 Contas a Pagar

O módulo de “Contas a Receber” tem a função de controlar e/ou fazer os pagamentos dos títulos oriundos das compras realizadas pela empresa. Além disto, é responsável por fazer qualquer outro pagamento de valores e efetuar o registro destes.

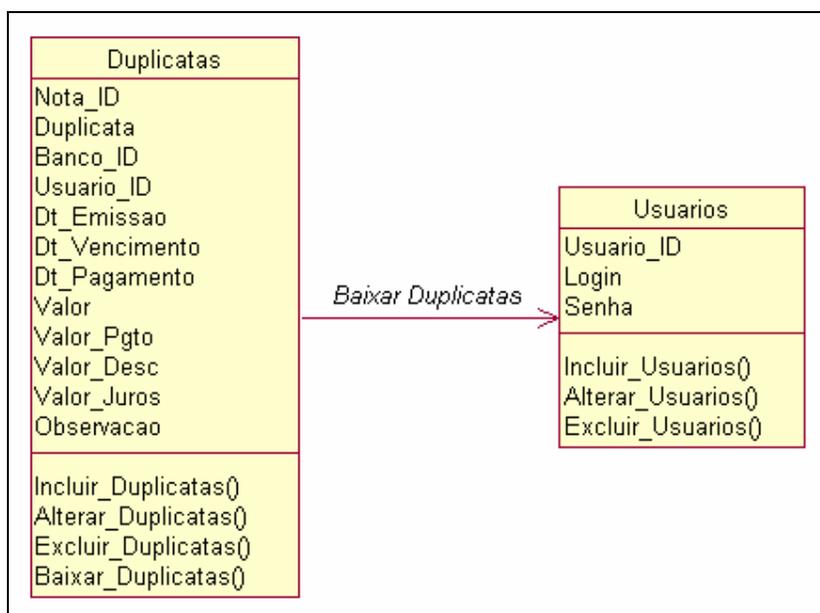
As duplicatas podem ser recebidas junto com a nota fiscal, como também podem vir via correio, através de um bloqueto (cobrança bancária). Assim que recebido o documento de cobrança, esta é programada para que o pagamento possa ser efetuado na data de vencimento.

Assim que o pagamento do título for efetuado, este será baixado no contas a pagar pelo usuário do sistema (figura 8.4).



**Figura 8.4 – Caso de Uso - Contas a Pagar**

No diagrama de classes da figura 8.5 percebe-se os relacionamentos entre as classes *Duplicatas* e *Usuarios*, onde o usuário do sistema realiza a baixa dos títulos que foram liquidados junto à empresa que emitiu a duplicata e/ou bloqueto.



**Figura 8.5 – Diagrama de Classes - Contas a Pagar**

No diagrama de seqüência da figura 8.6 está demonstrada a seqüência lógica das operações do contas a pagar. Neste diagrama percebe-se o fluxo “Baixar\_Duplicatas” que é realizado, pelo usuário do sistema, assim que um título for liquidado junto à empresa que emitiu a duplicata e/ou bloqueto.

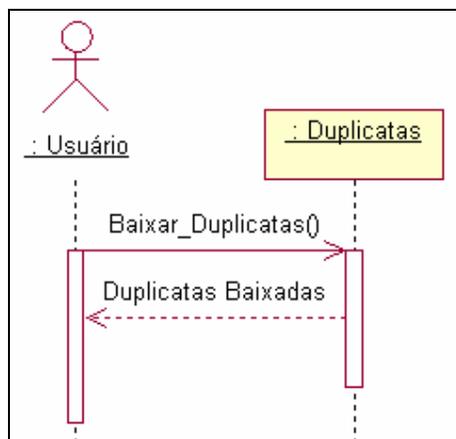


Figura 8.6 – Diagrama de Seqüência - Contas a Pagar

## 9 IMPLEMENTAÇÃO

Como já visto anteriormente, a arquitetura Multicamadas é uma evolução do modelo Cliente/Servidor, trazendo com ela inovações na arquitetura em relação à distribuição das aplicações, desempenho, segurança, escalabilidade e a confiabilidade dos sistemas, abrindo assim um novo leque de produtos.

A arquitetura multicamadas é um modelo de programação que prevê a divisão do programa em três ou mais partes bem definidas e distintas, que serão vistas com a implementação de dois módulos de um sistema de ERP, através das seguintes etapas:

- **Definição da tecnologia / estrutura:** nesta etapa realiza-se a escolha da linguagem de programação, do sistema gerenciador de banco de dados e seu acesso, da plataforma na qual o sistema deve ser executado, é definida a estrutura das camadas e de que forma serão implementadas as regras de negócio;
- **Mapeamento objeto-relacional:** optou-se pela utilização de um banco de dados relacional. Portanto, será necessário fazer o mapeamento objeto-relacional, onde as classes definidas nos diagramas de classes serão transformadas em entidades que vão compor o modelo entidade-relacionamento;
- **Implementação do software:** neste momento o *software* será efetivamente escrito na linguagem *Delphi* versão 8, utilizando os componentes SOAP da paleta WebServices. Para o acesso ao banco de dados, serão utilizados os componentes da paleta *DBExpress*.

## 9.1 Tecnologia / estrutura

O *software* foi desenvolvido na linguagem de programação Delphi versão 8.0, utilizando como meio de armazenamento o banco de dados *open source* Firebird<sup>16</sup>, versão 1.5. O *software* funciona no sistema operacional Windows, nas versões 98 ou superiores tendo instalado o IIS (*Internet Information Services*).

A estrutura do sistema foi implementada em três camadas, sendo a primeira a da apresentação, que é a camada da interface do cliente. A segunda, é a camada do servidor de aplicação, na qual estão as regras de negócio, sendo este um aplicativo executável que é executado pelo IIS. A terceira camada é da persistência, na qual está o servidor do banco de dados, neste caso o Firebird (figura 9.1).

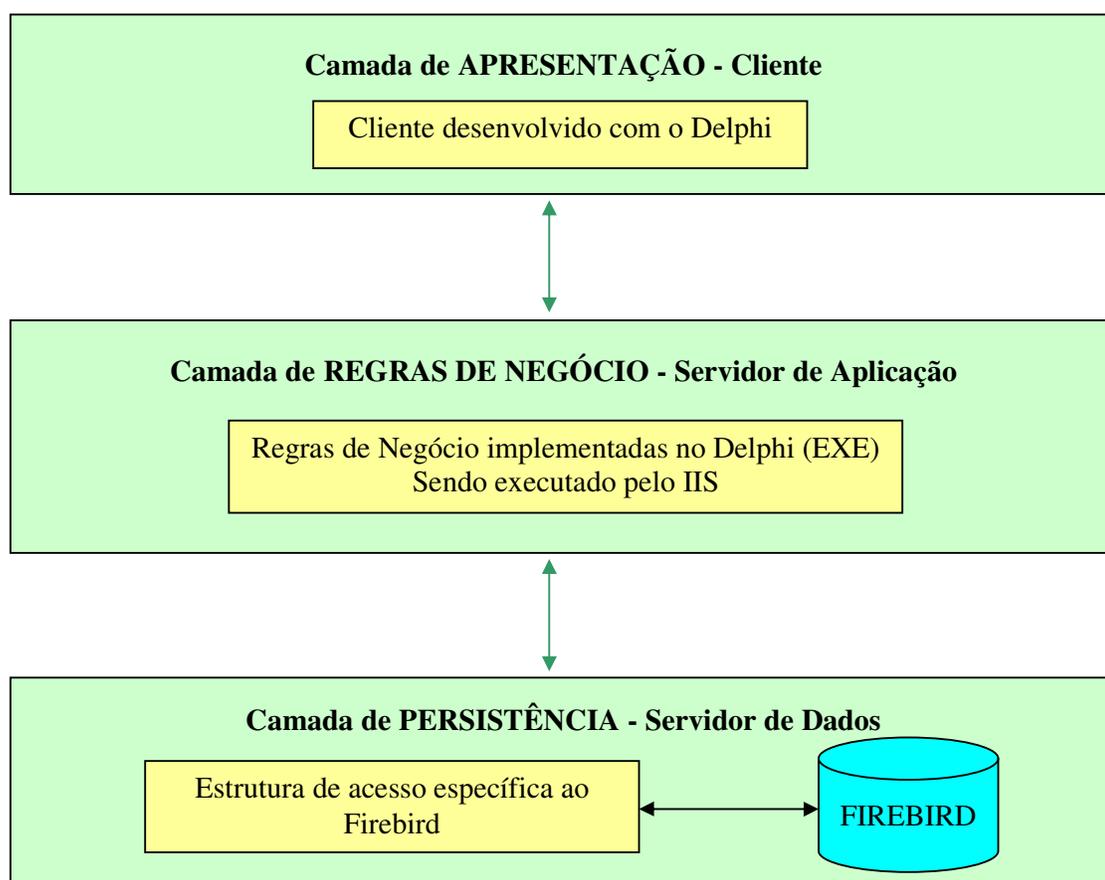


Figura 9.1 – Estrutura do Sistema ERP

<sup>16</sup> Disponível em < <http://www.firebirdsql.org> >. Acesso em 17 nov. 2004.

Para iniciar uma aplicação nCamadas usa-se o “SOAP Server Application” da aba WebServices. Esta opção cria o Servidor de Aplicação, no qual vão estar as regras de negócio do sistema (figura 9.2).

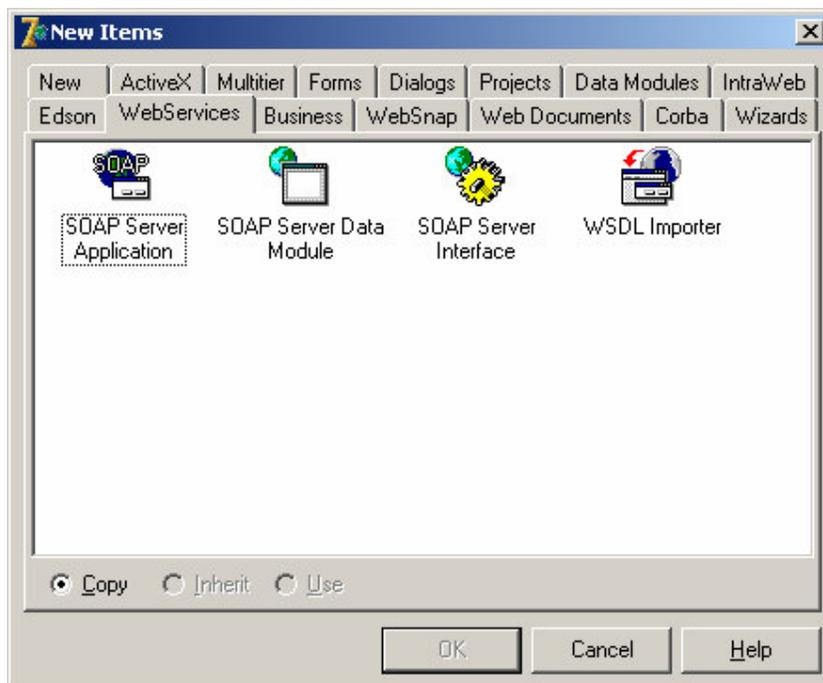


Figura 9.2 – Tela inicial para criação do Servidor de Aplicação

O próximo passo para a criação do Servidor de Aplicação deve ser a definição do tipo, podendo este ser uma DLL (*Dynamic Link Library*) ou um EXE (executável). Na criação deste projeto foi usado um executável (figura 9.3).

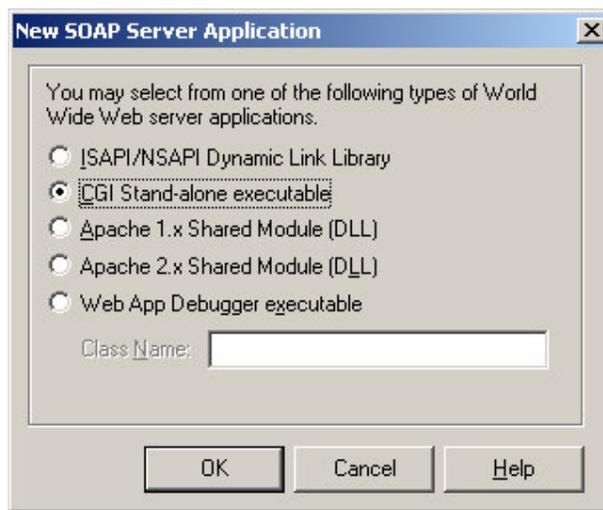


Figura 9.3 – Servidor de Aplicação tipo CGI (Executável)

Após a criação de Servidor de Aplicação, são criados os componentes do WebModule (figura 9.4). Estes componentes terão a função de publicar as informações do Servidor de Aplicação no padrão WSDL (*Web Service Description Language*), conforme já comentado no capítulo 4.



Figura 9.4 – Web Module do ERP

Em seguida, deve ser criado um *Data Module* (módulo de dados) do WebService (figura 9.5). Neste módulo de dados está o componente `SQLConnectionERP`, que contém todas as configurações e informações em relação ao banco de dados, tais como: tipo de banco (Firebird), localização, usuário, senha, etc.

Este módulo de dados também terá todos os componentes que irão realizar os cálculos sobre os dados. Um exemplo de um componente de cálculo é o `SQLdsCalculoNotas` (classe `TSQLDataSet`) que está ligado ao `DSPProviderCalculoNotas` (classe `TDataSetProvider`). É no componente `SQLdsCalculoNotas` que estão as instruções em relação ao cálculo da nota fiscal e o `DSPProviderCalculoNotas` tem a função de conectar a tabela à camada de apresentação da nota na qual, vai estar o componente da classe `TClientDataSet`.

Ainda neste *data module* tem-se todos os componentes que tem a função de incluir, alterar, excluir registros de uma tabela e pesquisar as informações. Como exemplo, têm-se os componentes `SQLdsClientes`, `SQLdsFornecedores`, `SQLdsPedidos`, sendo que cada um destes está conectado ao seu *provider* (figura 9.5).

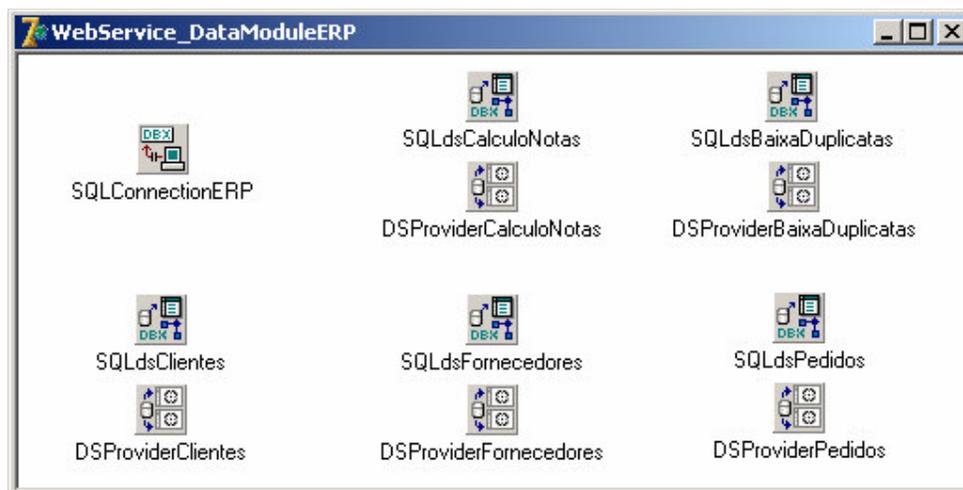


Figura 9.5 – Web Services - Data Module ERP

A figura 9.6 mostra o cadastro de clientes no modo de implementação e nesta figura pode-se ver o componente da classe `TClientDataSet` (comentado anteriormente) e o componente SOAP, que aponta a camada da apresentação para a URL que está sendo executada pelo IIS. A propriedade URL do SOAP, neste caso, vai estar assim: [http://SERVIDOR\\_APLICACAO/cgi-bin/ws.exe/soap/lwsdm](http://SERVIDOR_APLICACAO/cgi-bin/ws.exe/soap/lwsdm).

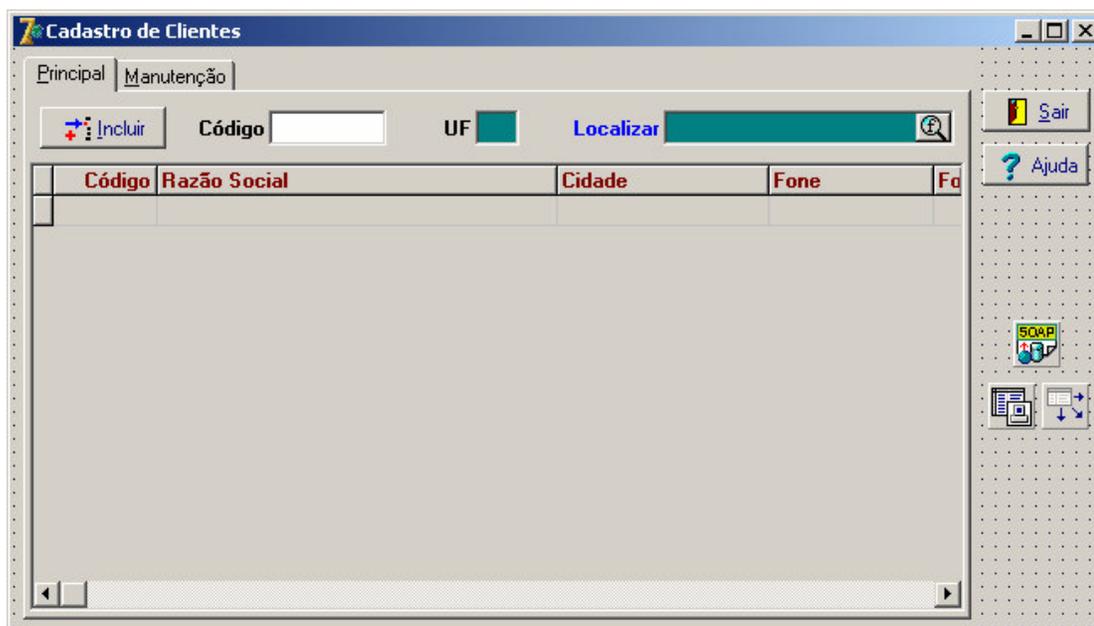


Figura 9.6 – Cadastro de Clientes - Modo de Implementação

Na figura 9.7 pode ser visto a configuração do aplicativo IIS, o qual tem a função de executar o aplicativo (Servidor de Aplicação) responsável pelas regras de negócio. O nome deste executável está como `ws.exe`, e se encontra na pasta “C:\Trabalho\Projeto\_Final\Servidor\cgi-bin”.

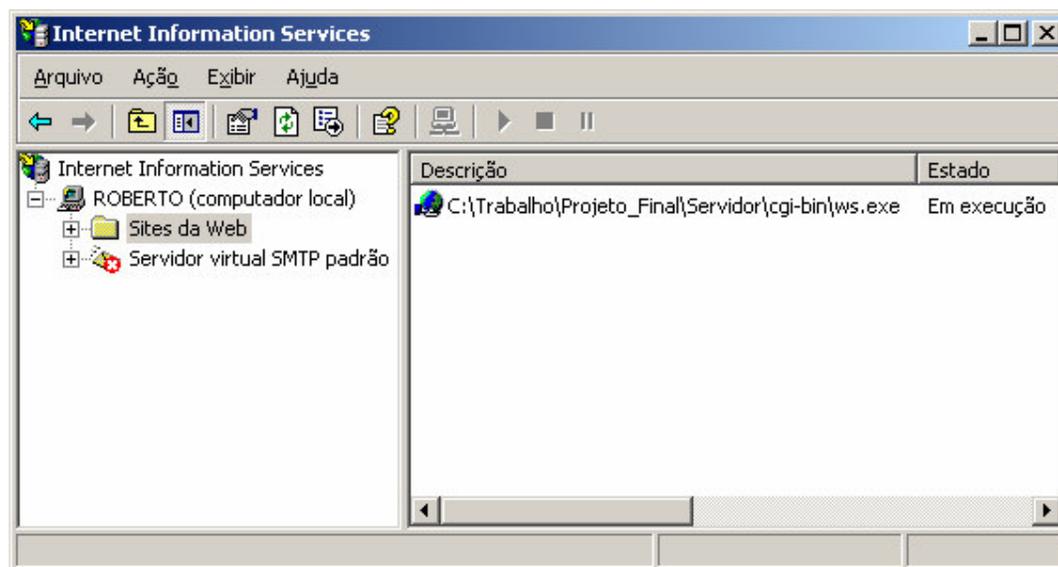


Figura 9.7 – Aplicativo IIS (*Internet Information Services*)

## 9.2 Mapeamento Objeto-Relacional

Para criação dos modelos de entidade-relacionamento foi utilizada a versão de demonstração do software ER/Studio versão 4.03, produzido pela empresa *Embarcadero Technologies*. Este software foi escolhido devido à sua facilidade de uso e ao fato de gerar os scripts para criação do banco de dados em vários formatos, inclusive para o padrão Firebird.

A figura 9.8 mostra o diagrama E-R criado para o mapeamento objeto-relacional do sistema. Neste diagrama estão representadas as chaves principais, as estrangeiras e os outros atributos de cada tabela.

Neste modelo as entidades principais (cor cinza) são as tabelas de: *Notas*, *Notas\_Itens*, *Pedidos*, *Pedidos\_Itens* e *Duplicatas*, que se relacionam através de *foreign keys* com várias entidades dentro do modelo. As entidades com a cor azul representam as tabelas relacionadas às pessoas e as entidades com as outras cores, são os cadastros auxiliares do sistema (figura 9.8).

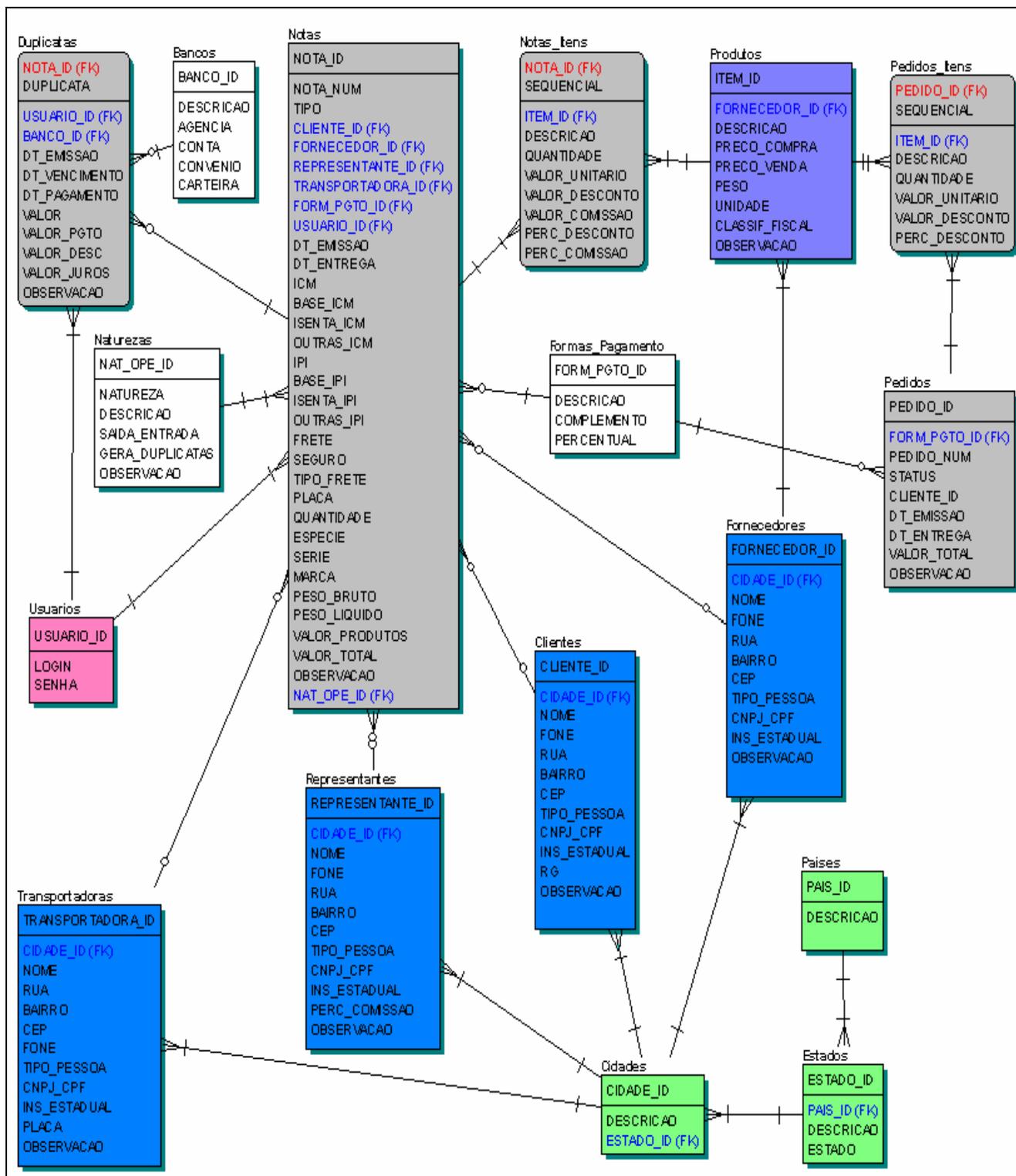


Figura 9.8 – Diagrama E-R do Faturamento / Financeiro

Nas tabelas abaixo estão representadas todas as entidades do sistema, identificadas pela chave primária (*primary key*), nome do atributo, tipo, se é requerido e suas chaves estrangeiras (*foreign key*). As entidades criadas para armazenar os cadastros básicos do sistema se encontram no final desta seção, tais como CLIENTES, FORNECEDORES, NATUREZAS, FORMAS\_PAGAMENTO, CIDADES, etc.

A tabela 9.1 mostra a estrutura da entidade PEDIDOS, utilizada para armazenar os dados do pedido feito pelo cliente. Nota-se nesta tabela a chave primária PEDIDO\_ID e as chaves estrangeiras CLIENTE\_ID e FORM\_PGTO\_ID das entidades CLIENTES e FORMAS\_PAGAMENTO respectivamente.

**Tabela 9.1 – Estrutura da entidade Pedidos**

PK	Campo	Tipo	Requerido	Foreign Key
✓	PEDIDO_ID	Integer	✓	CLIENTES FORMAS_PAGAMENTO
	PEDIDO_NUM	Varchar(20)	✓	
	CLIENTE_ID	Integer	✓	
	FORM_PGTO_ID	Integer	✓	
	STATUS	Char(1)		
	DT_EMISSAO	Date	✓	
	DT_ENTREGA	Date		
	VALOR_TOTAL	Numeric(18,4)		
	OBSERVAÇÃO	Text		

Logo em seguida, na tabela 9.2 tem-se a estrutura da entidade PEDIDOS\_ITENS, a qual é utilizada para armazenar os itens do pedido. Nota-se nesta tabela a chave primária composta por dois atributos PEDIDO\_ID e SEQUENCIAL e uma chave estrangeira ITEM\_ID da entidade PRODUTOS.

**Tabela 9.2 – Estrutura da entidade Pedidos\_Itens**

PK	Campo	Tipo	Requerido	Foreign Key
✓	PEDIDO_ID	Integer	✓	PRODUTOS
✓	SEQUENCIAL	Integer	✓	
	ITEM_ID	Varchar(20)	✓	
	DESCRICAÇÃO	Varchar(80)	✓	
	QUANTIDADE	Numeric(18,4)		
	VALOR_UNITARIO	Numeric(18,4)		
	VALOR_DESCONTO	Numeric(18,2)		
	PERC_DESCONTO	Numeric(9,2)		

A tabela 9.3 mostra a estrutura da entidade NOTAS, utilizada para armazenar os dados da emissão da nota fiscal. Nota-se nesta tabela a chave primária NOTA\_ID e diversas chaves estrangeiras as quais são necessárias dependendo do tipo da nota fiscal que está sendo emitida. Sendo as entidades das chaves estrangeiras as seguintes: CLIENTES, FORNECEDORES, REPRESENTANTES, TRANSPORTADORAS, FORMAS\_PAGAMENTO, NATUREZAS e USUARIOS.

**Tabela 9.3 – Estrutura da entidade Notas**

PK	Campo	Tipo	Requerido	Foreign Key
✓	NOTA_ID	Integer	✓	
	NOTA_NUM	Integer	✓	
	TIPO	Char(1)	✓	
	CLIENTE_ID	Integer		CLIENTES
	FORNECEDOR_ID	Integer		FORNECEDORES
	REPRESENTANTE_ID	Integer		REPRESENTANTES
	TRANSPORTADORA_ID	Integer		TRANSPORTADORAS
	FORM_PGTO_ID	Integer		FORMAS_PAGAMENTO
	NAT_OPE_ID	Integer	✓	NATUREZAS
	USUARIO_ID	Integer	✓	USUARIOS
	DT_EMISSAO	Date	✓	
	DT_ENTREGA	Date	✓	
	ICM	Numeric(18,2)		
	BASE_ICM	Numeric(18,2)		
	ISENTA_ICM	Numeric(18,2)		
	OUTRAS_ICM	Numeric(18,2)		
	IPI	Numeric(18,2)		
	BASE_IPI	Numeric(18,2)		
	ISENTA_IPI	Numeric(18,2)		
	OUTRAS_IPI	Numeric(18,2)		
	FRETE	Numeric(18,2)		
	SEGURO	Numeric(18,2)		
	TIPO_FRETE	Char(1)		
	PLACA	Varchar(12)		
	QUANTIDADE	Numeric(9,2)		
	ESPECIE	Varchar(12)		
	SERIE	Varchar(12)		
	MARCA	Varchar(12)		
	PESO_BRUTO	Numeric(9,2)		
	PESO_LIQUIDO	Numeric(9,2)		
	VALOR_PRODUTOS	Numeric(18,4)		
	VALOR_TOTAL	Numeric(18,4)		
	OBSERVACAO	Text		

A tabela 9.4 mostra a estrutura da entidade NOTAS\_ITENS, utilizada para armazenar os itens da nota fiscal.. Nesta tabela têm-se os atributos NOTA\_ID e SEQUENCIAL que formam a chave primária e a chave estrangeira ITEM\_ID da entidade PRODUTOS.

**Tabela 9.4 – Estrutura da entidade Notas\_Itens**

PK	Campo	Tipo	Requerido	Foreign Key
✓	NOTA_ID	Integer	✓	PRODUTOS
✓	SEQUENCIAL	Integer	✓	
	ITEM_ID	Varchar(20)	✓	
	DESCRICAO	Varchar(80)		
	QUANTIDADE	Numeric(18,4)		
	VALOR_UNITARIO	Numeric(18,4)		
	VALOR_DESCONTO	Numeric(18,2)		
	VALOR_COMISSAO	Numeric(18,2)		
	PERC_DESCONTO	Numeric(9,2)		
	PERC_COMISSAO	Numeric(9,2)		

Na tabela 9.5 tem-se a representação da entidade DUPLICATAS, a qual é utilizada para armazenar as duplicatas originadas pela forma de pagamento da nota fiscal.. Nesta tabela tem-se a chave primária NOTA\_ID e as chaves estrangeiras USUARIO\_ID e BANCO\_ID, respectivamente das entidades USUARIOS e BANCOS.

**Tabela 9.5 – Estrutura da entidade Duplicatas**

PK	Campo	Tipo	Requerido	Foreign Key
✓	NOTA_ID	Integer	✓	USUARIOS BANCOS
	DUPLICATA	Varchar(2)	✓	
	USUARIO_ID	Integer	✓	
	BANCO_ID	Integer		
	DT_EMISSAO	Date		
	DT_VENCIMENTO	Date		
	DT_PAGAMENTO	Date		
	VALOR	Numeric(18,2)		
	VALOR_PGTO	Numeric(18,2)		
	VALOR_DESC	Numeric(18,2)		
	VALOR_JUROS	Numeric(18,2)		
	OBSERVACAO	Text		

Nas tabelas abaixo se encontram todas as outras entidades usadas nos módulos do Faturamento e Financeiro do sistema de ERP.

**Tabela 9.6 – Estrutura da entidade Clientes**

PK	Campo	Tipo	Requerido	Foreign Key
✓	CLIENTE_ID	Integer	✓	CIDADES
	CIDADE_ID	Integer	✓	
	NOME	Varchar(50)	✓	
	FONE	Varchar(15)		
	RUA	Varchar(50)	✓	
	BAIRRO	Varchar(30)		
	CEP	Varchar(10)		
	TIPO_PESSOA	Char(1)	✓	
	CNPJ_CPF	Varchar(20)	✓	
	INS_ESTADUAL	Varchar(15)		
	RG	Varchar(15)		
	OBSERVACAO	Text		

**Tabela 9.7 – Estrutura da entidade Fornecedores**

PK	Campo	Tipo	Requerido	Foreign Key
✓	FORNECEDOR_ID	Integer	✓	CIDADES
	CIDADE_ID	Integer	✓	
	NOME	Varchar(50)	✓	
	FONE	Varchar(15)		
	RUA	Varchar(50)	✓	
	BAIRRO	Varchar(30)		
	CEP	Varchar(10)		
	TIPO_PESSOA	Char(1)	✓	
	CNPJ_CPF	Varchar(20)	✓	
	INS_ESTADUAL	Varchar(15)		
	OBSERVACAO	Text		

**Tabela 9.8 – Estrutura da entidade Representantes**

PK	Campo	Tipo	Requerido	Foreign Key
✓	REPRESENTANTE_ID	Integer	✓	CIDADES
	CIDADE_ID	Integer	✓	
	NOME	Varchar(50)	✓	
	FONE	Varchar(15)		
	RUA	Varchar(50)	✓	
	BAIRRO	Varchar(30)		
	CEP	Varchar(10)		
	TIPO_PESSOA	Char(1)	✓	
	CNPJ_CPF	Varchar(20)	✓	
	PERC_COMISSAO	Numeric(9,2)		
	OBSERVACAO	Text		

Tabela 9.9 – Estrutura da entidade Transportadoras

PK	Campo	Tipo	Requerido	Foreign Key
✓	TRANSPORTADORA_ID	Integer	✓	CIDADES
	CIDADE_ID	Integer	✓	
	NOME	Varchar(50)	✓	
	FONE	Varchar(15)		
	RUA	Varchar(50)	✓	
	BAIRRO	Varchar(30)		
	CEP	Varchar(10)	✓	
	TIPO_PESSOA	Char(1)	✓	
	CNPJ_CPF	Varchar(20)	✓	
	PLACA	Varchar(12)		
	OBSERVACAO	Text		

Tabela 9.10 – Estrutura da entidade Produtos

PK	Campo	Tipo	Requerido	Foreign Key
✓	ITEM_ID	Integer	✓	FORNECEDORES
	DESCRICAO	Varchar(80)	✓	
	FORNECEDOR_ID	Integer	✓	
	PRECO_COMPRA	Numeric(18,4)		
	PRECO_VENDA	Numeric(18,4)		
	PESO	Numeric(9,2)		
	UNIDADE	Char(2)		
	CLASSIF_FISCAL	Varchar(20)		
	OBSERVACAO	Text		

Tabela 9.11 – Estrutura da entidade Naturezas

PK	Campo	Tipo	Requerido	Foreign Key
✓	NAT_OPE_ID	Integer	✓	
	NATUREZA	Varchar(10)	✓	
	DESCRICAO	Varchar(50)	✓	
	SAIDA_ENTRADAS	Char(1)	✓	
	GERA_DUPLICATAS	Char(1)	✓	
	OBSERVACAO	Text		

Tabela 9.12 – Estrutura da entidade Formas\_Pagamento

PK	Campo	Tipo	Requerido	Foreign Key
✓	FORM_PGTO_ID	Integer	✓	
	DESCRICAO	Varchar(50)	✓	
	COMPLEMENTO	Varchar(20)		
	PERCENTUAL	Numeric(12)		

Tabela 9.13 – Estrutura da entidade Bancos

PK	Campo	Tipo	Requerido	Foreign Key
✓	BANCO_ID DESCRICAO AGENCIA CONTA CONVENIO CARTEIRA	Integer Varchar(50) Varchar(15) Varchar(15) Varchar(15) Varchar(3)	✓ ✓	

Tabela 9.14 – Estrutura da entidade Usuarios

PK	Campo	Tipo	Requerido	Foreign Key
✓	USUARIO_ID LOGIN SENHA	Integer Varchar(20) Varchar(10)	✓ ✓ ✓	

Tabela 9.15 – Estrutura da entidade Cidades

PK	Campo	Tipo	Requerido	Foreign Key
✓	CIDADE_ID DESCRICAO ESTADO_ID	Integer Varchar(40) Integer	✓ ✓ ✓	ESTADOS

Tabela 9.16 – Estrutura da entidade Estados

PK	Campo	Tipo	Requerido	Foreign Key
✓	ESTADO_ID DESCRICAO ESTADO PAIS_ID	Integer Varchar(40) Char(2) Integer	✓ ✓ ✓ ✓	PAISES

Tabela 9.17 – Estrutura da entidade Paises

PK	Campo	Tipo	Requerido	Foreign Key
✓	PAIS_ID DESCRICAO	Integer Varchar(40)	✓ ✓	

### 9.3 Implementação do Software

Nas seções a seguir tem-se a implementação das três camadas: (i) Regras de Negócio; (ii) Apresentação e (iii) Persistência. Estas foram implementadas nos módulos do Faturamento e do Financeiro do sistema de ERP.

#### 9.3.1 Regras de Negócio

A camada das Regras de Negócio, como já visto anteriormente, contém todas as informações em relação à “lógica” do sistema. Também nesta camada é determinado como os dados serão utilizados pela camada da Apresentação.

A figura 9.9 mostra o Data Module ERP, neste estão os componentes que contém as regras utilizadas nos módulo do Faturamento e do Financeiro.

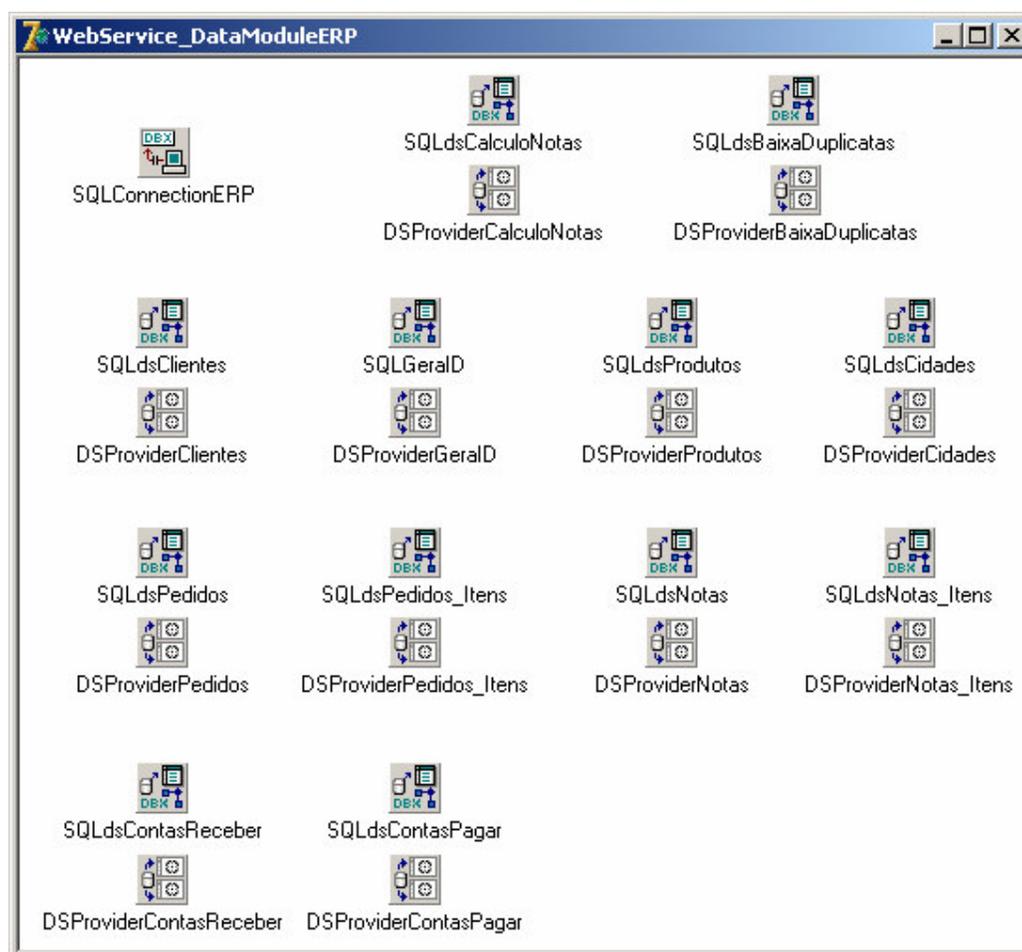


Figura 9.9 – Web Services - Data Module ERP

Na figura 9.9 tem-se o componente `SQLdsClientes` que está associado ao componente `DSPProviderClientes`. No evento `OnUpdateError` do `DSPProviderClientes` está definida a regra “Valida CPF” a qual está descrita na figura 9.10. Esta regra testa a existência do CPF.

```
if pos('UK_CPF',UpperCase(E.Message)) > 0 then
begin
  raise EDatabaseError.Create('Número de CPF ' +
    String(Dataset.FieldValues['CPF']) + já existe !)
end
else
begin
  raise EDatabaseError.Create('Ocorreu o seguinte erro:' + E.Message);
end;
```

Figura 9.10 – Regra de Negócio - Valida CPF

Outro exemplo de regra de negócio pode ser visto na figura 9.11. Esta regra gera um código seqüencial para a tabela de clientes e está associada aos componentes `SQLGeraID` e `DSPProviderGeraID`. A regra está definida no evento do `BeforeUpdateRecord` do `DSPProviderGeraID`.

```
if UpdateKind = ukInsert then
begin
  sqlGeraID.Close;
  sqlGeraID.CommandText :=
    'SELECT gen_id(GEN_CLIENTES_ID,1)' +
    ' as CLIENTE_ID from RDB$DATABASE';
  sqlGeraID.Open;
end;
```

Figura 9.11 – Regra de Negócio - Gera ID

### 9.3.2 Apresentação

Na camada da Apresentação (aplicativo das estações), também conhecida como Interface, permanecem todas as telas, pelas quais o usuário interage com o sistema.

Para esta camada, optou-se pela criação de uma tela inicial com atalhos (ícones) para as principais operações disponíveis, sempre visando a simplicidade do sistema. Foram criados atalhos para os cadastros principais e para os módulos do faturamento e do financeiro. Os outros atalhos foram adicionados para facilitar o uso do sistema. Logo acima dos ícones tem-se o menu do sistema completo, conforme mostra a figura 9.12, e um pouco abaixo aparece o nome do usuário (USUÁRIO DO SISTEMA) que está logado no sistema.

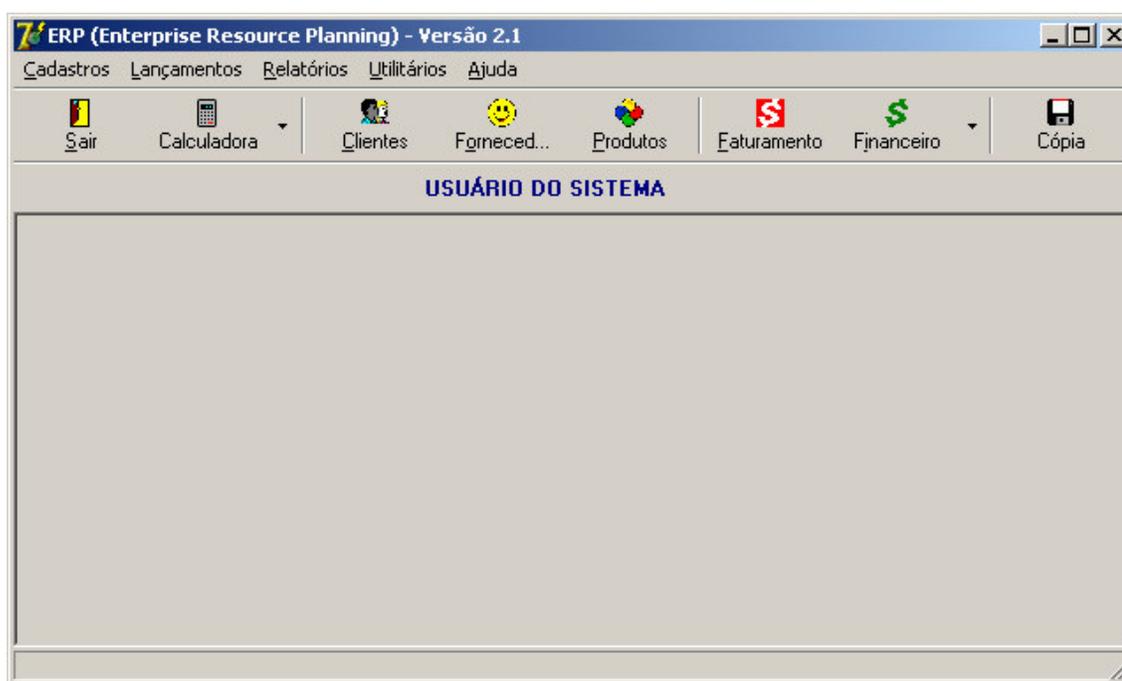


Figura 9.12 – ERP - Tela Inicial

#### A) Cadastros

Todos os cadastros seguem um padrão de funcionamento. A tela do cadastro de clientes será usada como base para descrever a utilização do sistema. Ao se clicar no menu para entrar na tela correspondente, o sistema abre a mesma em modo de pesquisa. Neste modo, todos os registros da tabela correspondente são exibidos, um abaixo do outro, com a descrição e as observações constantes em cada registro. Para localizar-se um registro pode-se usar o campo `Código` digitando o código do registro ou usar a opção `Localizar` (figura 9.13).

O botão **Incluir** tem a função de fazer a entrada dos dados através da tela de **Manutenção** (figura 9.14). Após a digitação dos dados estes devem ser gravados através do botão **Grava**. Através de um duplo click sobre um registro da tela **Principal** passa-se para a tela de **Manutenção** e é neste momento que os registros podem ser alterados ou excluídos pressionando os botões **Grava** e **Exclui** respectivamente.

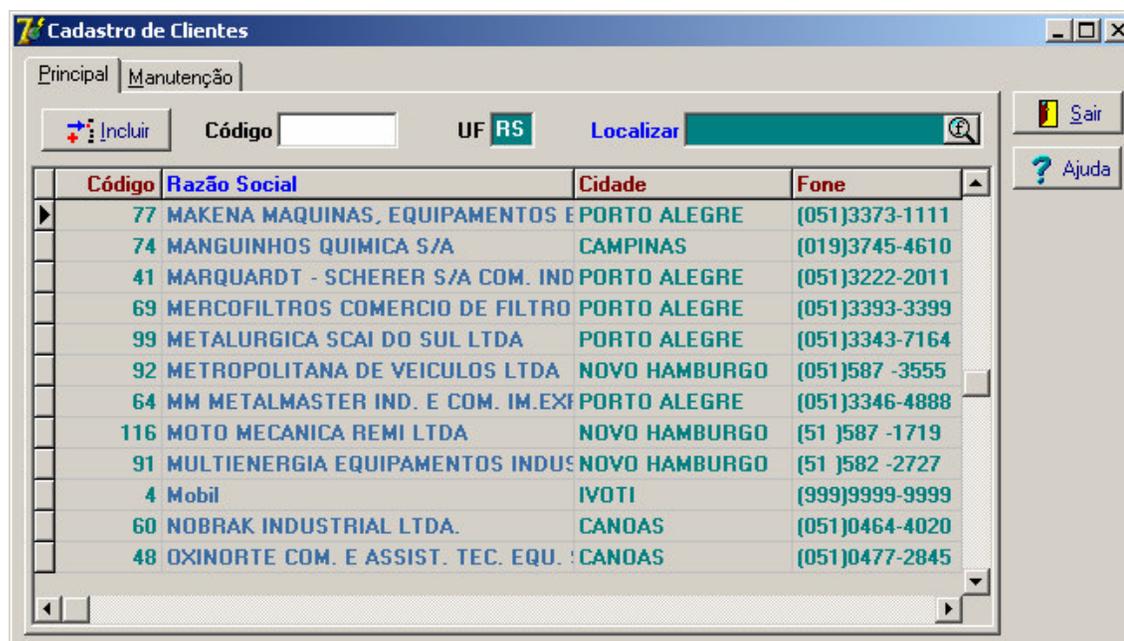


Figura 9.13 – Tela Principal do Cadastro de Clientes



Figura 9.14 – Tela de Manutenção do Cadastro de Clientes

## B) Pedidos

A figura 9.15 mostra a tela `Principal` dos pedidos, a qual tem um modo de funcionamento semelhante ao cadastro de clientes.

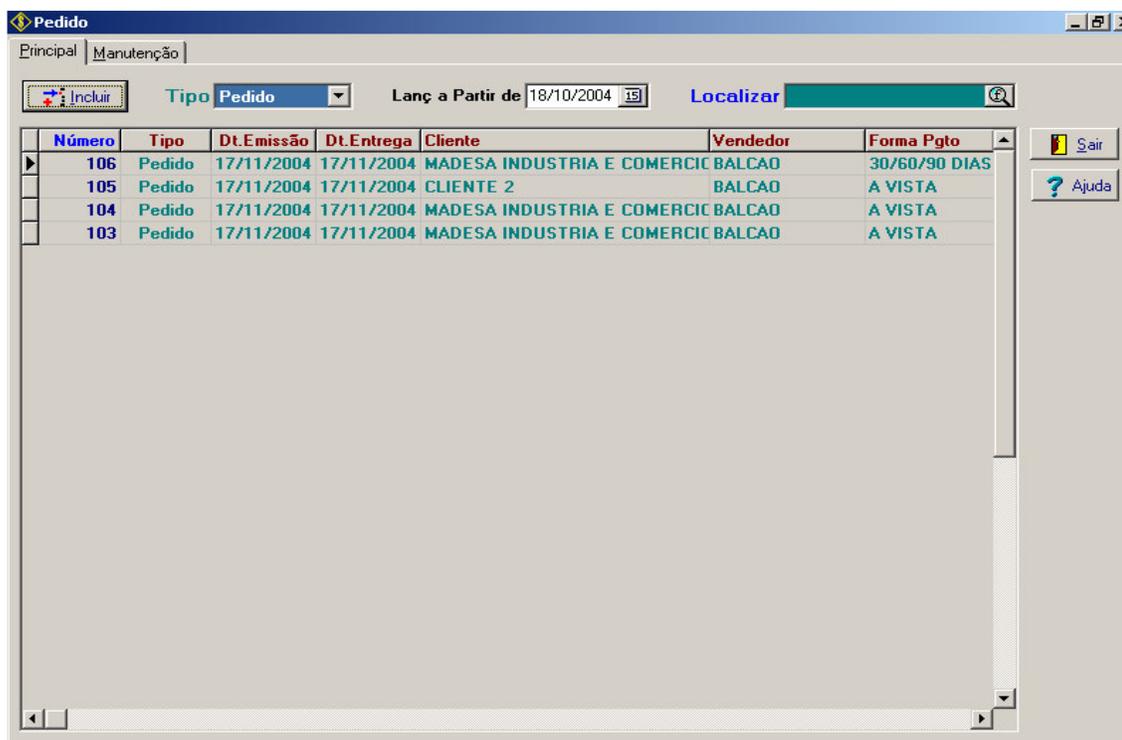


Figura 9.15 – Tela Principal dos Pedidos

A inclusão do pedido será feita pelo cliente ou pelo usuário do sistema. Conforme mostra a figura 9.16 os seguintes dados devem ser informados: nome do cliente, vendedor e os produtos do pedido (lado direito da tela) e logo abaixo a forma de pagamento. No lado esquerdo da tela ainda tem-se dos dados do cliente. Caso estejam desatualizados, estes podem ser alterados. Além disto, no canto inferior esquerdo, tem-se os dados das últimas compras do cliente, servindo como um histórico do cliente.

**Cliente** MADESA INDUSTRIA E COMERCIO **Vendedor** BALCAO - (1) **Tipo** Pedido

**Fone** [ ] **Ped: 106** **Cód. Cliente** 3 **Emissão** 17/11/2004 **Entrega** 17/11/2004

**Dados do Cliente**

**Fone / Fax** (051) 594-1009 **Dt. Atualiz** [ ]

**CNPJ** 04.115.990/0001-64 **Contato** José

**Logradouro**

**Endereço do Cliente** [ ] **Bairro** [ ]

**Cidade** IVOTI **CEP** 93900-000 **RS**

**Local de Entrega**

**Endereço de Entrega Cliente** [ ] **Bairro** [ ]

**Cidade** IVOTI **CEP** 93900-000 **RS**

**Últimos Lançamentos**

**Inicial** 18/09/2004 **Final** 16/01/2005

**Documento** Geral **Pesquisar** [ ]

Dt. Emissão	Código	Produto	Quantid	Vlr. Unit	% Ds
17/11/2004	1	Produto 1	1,00	100,00	0,00
17/11/2004	2	Produto 2	1,00	200,00	0,00
17/11/2004	1	Produto 1	1,00	100,00	0,00
17/11/2004	1	Produto 1	1.000,00	10.000,00	0,00
29/09/2004	1	Produto 1	1,00	100,00	10,00
29/09/2004	2	Produto 2	1,00	200,00	0,00
29/09/2004	1	Produto 1	1,00	100,00	0,00
29/09/2004	2	Produto 2	1,00	200,00	0,00

**Produtos / Itens**

Código	Produto	Quantid	Valor Unt	% Ds
1	Produto 1	1,00	100,00	0,00
2	Produto 2	1,00	200,00	0,00

**Desconto** R\$ 0,00 **Total** R\$ 300,00

**Pagamento** 30/60/90 DIAS - (6)

**Cl. Fiscal:** 1-12345

**Exclui** **Cancela** **Grava** **Pedido**

**Duplicatas** **Pesquisar Documentos**

Duplicatas	Vencimento	Valor	Pagamento	Valor Pago	Desconto	Juros

Figura 9.16 – Tela de Manutenção dos Pedidos

### C) Notas Fiscais

A figura 9.17 mostra a tela *Principal* da emissão das notas fiscais, a qual também segue o padrão de funcionamento do sistema. Para a emissão de uma nota fiscal sem pedido, deve-se pressionar o botão *Incluir* e preencher os dados na tela de *Manutenção* (figura 9.18).

Para emitir uma nota fiscal referente a um pedido, estes devem ser selecionados através da opção *Procura* e o resultado desta pesquisa será exibido na tela *Principal*. Na tela *Principal* clicando sobre um pedido se passará para a tela de *Manutenção* na qual os dados do pedido devem ser conferidos e em seguida a nota fiscal será emitida (figura 9.18).

**Emissão das Notas Fiscais**

Acesso | Manutenção | Pesquisa

Incluir Procura Nota Lanç a Partir de 21/07/2003 Saída Entrada Sair Ajuda

Número	Dt.Emissão	Dt.Entrega	Valor Total	% Comis	% ICM	Valor ICM	Base ICM	Valor IPI
77	29/09/2004	29/09/2004	300,00	0,00	17,00	51,00	300,00	0,00
76	29/09/2004		290,00	0,00	17,00	49,30	290,00	0,00
74	10/09/2004		100,00	0,00	17,00	18,70	110,00	0,00
73	10/09/2004		201,31	0,00	17,00	38,03	223,68	0,00
72	10/09/2004		90,00	0,00	17,00	17,00	100,00	0,00
71	08/09/2004		101,00	0,00	17,00	17,17	101,00	0,00
70	25/08/2004	24/09/2004	330,00	0,00	17,00	56,10	330,00	0,00
69	25/08/2004	24/09/2004	300,00	0,00	17,00	51,00	300,00	0,00
68	23/07/2004	23/07/2004	8.300,00	0,00	17,00	1.411,00	8.300,00	0,00
66	23/07/2004	23/07/2004	83,00	0,00	17,00	14,11	83,00	0,00
65	23/07/2004		1.411,00	0,00	17,00	239,87	1.411,00	0,00
64	23/07/2004		83,00	0,00	17,00	14,11	83,00	0,00
61	05/05/2004		50,00	0,00	17,00	8,50	50,00	0,00
59	30/03/2004		50,00	0,00	17,00	8,50	50,00	0,00
58	30/03/2004		11,00	0,00	17,00	1,87	11,00	0,00
57	06/02/2004		1.810,00	0,00	17,00	307,70	1.810,00	0,00
56	06/02/2004		450,00	0,00	17,00	76,50	450,00	0,00
55	06/02/2004		14,00	0,00	17,00	2,38	14,00	0,00
54	02/02/2004		11,00	0,00	17,00	1,87	11,00	0,00
53	30/12/2003		150,00	0,00	17,00	23,80	140,00	10,00
52	22/12/2003		104,00	0,00	17,00	17,68	104,00	0,00

Ordem Cidade do Cliente Consulta  Dados Adicionais

**Figura 9.17 – Tela Principal da Emissão de Nota Fiscal**

Na figura 9.18, conforme já mencionada, os dados devem ser preenchidos e a impressão (emissão) da nota fiscal será realizada clicando-se no botão *Imprime*. Nesta tela ainda tem-se a aba de *Pesquisa* que realiza uma pesquisa mais completa sobre os dados e também o botão *Duplicatas*, que mostrará as duplicatas referentes à nota.

**Emissão das Notas Fiscais**

Principal | Manutenção | Pesquisa

Nota Fiscal: 77      Pedido: 1212..       Cliente  
 Fornecedor

Emissão: 15/11/2004      Data / Hora Saída:      ? Ajuda

Cliente: ADENAR NESTOR KOCK      Nat. Op: 5102      Venda de mercadoria adquirida ou re

Represent: BALCAO      Comissão: 0 %      ICMS: 17 %      Trsf. ICMS: 00,00 %

Código	Produto	Quantidade	Valor Unitário	% ICM	% IPI	Valor IPI	% Desc	% Comis
1	Produto 1	1,00	100,00	17,00	00,00	0,00	00,00	00,00
2	Produto 2	1,00	200,00	17,00	00,00	0,00	00,00	00,00

Base ICMS: 300,00      ICMS: 51,00      ICMS Subst: 0,00      Tot. Produtos: 300,00

Valor Frete: 0,00      Seguro: 0,00      Valor IPI: 0,00      Total Nota: 300,00

Transport: PROPRIO      Frete por Conta:  Emitente       Destinatário      Placa: IDV7890      Cidade: IVOTI

Quantidade: 0,00      Espécie:      Marca:      Peso Bruto: 15,00      Peso Líquido: 10,00

OBSERVÇ: Pedido: 1212      Forma Pgto: 30/60/90 DIAS      Praça Pgto: IVOTI  
Representante: FLAVIO      Observação Adicional:       Duplicatas

Grava      Cancela      Exclui      Imprime

Figura 9.18 – Tela de Manutenção da Emissão de Nota Fiscal

#### D) Contas a Receber

A tela do contas a receber também segue o padrão de funcionamento do sistema, tendo a tela Principal (figura 9.19) e a de Manutenção (figura 9.20).

Na tela Principal tem-se a opção de selecionar os títulos abertos, pagos ou ambos. na tela de Manutenção é efetuada a baixa dos títulos pagos pelos clientes e nesta também se consegue ver os títulos baixados automaticamente através do retorno do banco.

Contas a Receber

Principal | Manutenção

Incluir Duplicata

Duplicatas:  Abertas  Pagas  Ambas

Sair Ajuda

Duplicata	s	Pedido	Cliente	Pagamento	Vencimento	Emissão	Valor
37	A		MADESA INDUSTRIA E		15/09/2003	05/09/2003	500,00
37	B		MADESA INDUSTRIA E	10/10/2001	25/09/2003	05/09/2003	500,00
35	A		MADESA INDUSTRIA E		05/09/2003	05/09/2003	0,00
34	A		MADESA INDUSTRIA E		05/09/2003	05/09/2003	0,00
33	A		MADESA INDUSTRIA E		04/09/2003	04/09/2003	35,00
32	A		MADESA INDUSTRIA E		04/09/2003	04/09/2003	15,00
31	A		MADESA INDUSTRIA E		04/09/2003	04/09/2003	1,00
30	A		MADESA INDUSTRIA E		04/09/2003	04/09/2003	5,00
28	A		MADESA INDUSTRIA E		04/09/2003	04/09/2003	15,80
27	A		MADESA INDUSTRIA E	20/12/2001	04/09/2003	04/09/2003	5,50
26	A		MADESA INDUSTRIA E	10/10/2001	04/09/2003	04/09/2003	1,10
25	A		MADESA INDUSTRIA E	10/10/2001	04/09/2003	04/09/2003	1,10
24	A		MADESA INDUSTRIA E	10/10/2001	04/09/2003	04/09/2003	41,90
23	A		MADESA INDUSTRIA E		04/09/2003	04/09/2003	3,50
22	A		MADESA INDUSTRIA E	10/10/2001	04/09/2003	04/09/2003	5,90
21	A		MADESA INDUSTRIA E	10/10/2001	04/09/2003	04/09/2003	9,90
20	A		MADESA INDUSTRIA E	03/09/2003	04/09/2003	04/09/2003	9,90

Ordem: Duplicata Consulta

Figura 9.19 – Tela Principal do Contas a Receber

Contas a Receber

Principal | Manutenção

Sair Ajuda

Duplicata: 37 B Cliente: MADESA INDUSTRIA E COMERCIO

Data Pagamento: 10/10/2001 Data Vencimento: 25/09/2003 Data Emissão: 05/09/2003

Valor: 500,00 Multa: % Desconto: 0,00 %

Valor Pago: 500,00 Honorários: % Juros: 0,00 %

Histórico

Forma Pagto: Bloqueto Código

Banco: BCN Carteira / Descont:   Núm. Banco:

Duplicata	s	Pagamento	Vencimento	Emissão	Valor	Valor Pago	Desconto
37	A		15/09/2003	05/09/2003	500,00		0,00
37	B	10/10/2001	25/09/2003	05/09/2003	500,00	500,00	0,00

Grava Cancela Exclui Duplicata Bloqueto Recibo

Figura 9.20 – Tela de Manutenção do Contas a Receber

### 9.3.3 Persistência

A camada de Persistência fica no servidor do banco de dados, na qual reside toda informação necessária para o funcionamento da aplicação e os dados do sistema. Cabe ressaltar, que os dados somente são acessados através do Servidor de Aplicação, e não diretamente pela aplicação do cliente (estação).

No *Data Module* (figura 9.21) tem-se o componente *SQLConnectionERP* pelo qual é feito a conexão com o banco de dados. Neste componente também são feitas todas as outras configurações e informações que dizem respeito ao banco de dados, tais como: *database* (local do banco), *login*, *password*, etc.

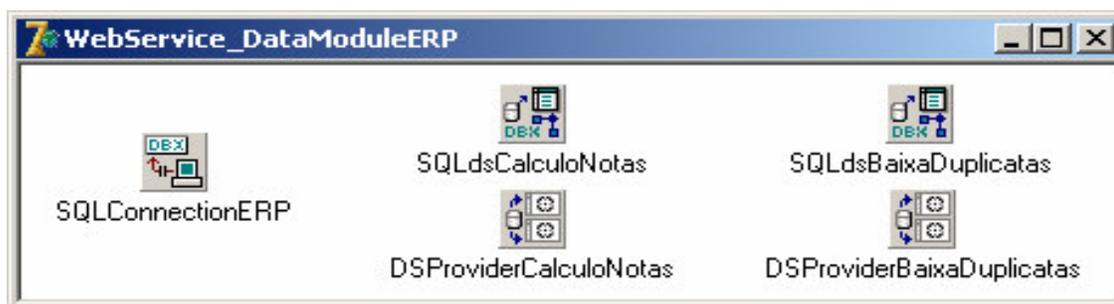


Figura 9.21 – Web Services - Data Module ERP

## CONCLUSÃO

A automatização do ambiente empresarial, a integração de tarefas, o aumento do número de usuários, a atualização constante dos sistemas e em diferentes pontos da rede, têm afetado de forma prejudicial o desempenho, a segurança, a escalabilidade e a confiabilidade dos sistemas. Neste contexto, a arquitetura de sistemas Multicamadas surge como uma alternativa para minimizar estes problemas.

A arquitetura multicamadas é um modelo de programação que prevê a divisão do programa em três ou mais partes bem definidas e distintas, sendo que as principais são: Interface (apresentação), Regras de Negócio (lógica) e Persistência (banco de dados). Com a introdução da camada de lógica, as Regras de Negócio concentram-se no Servidor de Aplicações. Como são nestas regras que ocorre a maior partes das mudanças e uma vez que elas estão centralizadas no Servidor de Aplicações, resolve-se o problema da atualização, em centenas ou milhares de computadores, cada vez que uma Regra de Negócio for alterada, facilitando em muito a atualização dos sistemas.

Ao longo dos anos, ferramentas têm sido criadas para agilizar, facilitar e melhorar a montagem dos projetos e o desenvolvimento das aplicações. Com estas ferramentas, tornou-se indispensável à utilização de componentes, bibliotecas e padrões de projeto visando à produtividade no processo de desenvolvimento dos *softwares* e a aceitação dos mesmos, ainda mais neste mercado que é altamente competitivo. Neste contexto surge o Delphi, que é uma ferramenta que possui diversas bibliotecas de componentes para o desenvolvimento de sistemas Multicamadas.

A UML define uma série de diagramas com objetivo de facilitar a forma como são representados projetos de software. Pode-se perceber na prática os benefícios gerados a partir da utilização da UML, dentre os quais destacam-se: a fácil compreensão dos modelos criados e melhor representação visual do sistema, gerando um projeto melhor definido.

O estudo de caso com dois módulos de um software ERP permitiu verificar que: (i) a utilização do Delphi em conjunto com o SOAP e os *Web Services* facilitam a implementação de sistemas multicamadas; (ii) é extremamente positiva a separação das regras de negócio das outras camadas, pois torna mais fácil a manutenção e a atualização do sistema e contribui com a segurança do sistema, uma vez que o cliente (apresentação) não tem acesso direto aos dados (persistência); (iii) as aplicações clientes se tornam menores, pois o processamento (regras de negócio) é realizado na camada do servidor de aplicação; (iv) com a alocação de desenvolvedores para a construção de camadas específicas, estas podem ser desenvolvidas simultaneamente, tornando os desenvolvedores especialistas e podendo assim, as equipes trabalhar exclusivamente para cada camada do sistema.

Como trabalhos futuros, sugere-se o projeto e implementação dos outros módulos do sistema ERP e a realização de testes em relação a velocidade de processamento, pois esta velocidade poderá ser melhor testada e avaliada com todos os módulos de um sistema de ERP e com o acesso simultâneo de diversos usuários.

## REFERÊNCIAS BIBLIOGRÁFICAS

- ARAÚJO, Edvar Bergmann. **DOMonitor Um Ambiente de Monitoração de Aplicações Distribuídas Java**. Porto Alegre: UGRGS, 2001.
- BATTISTI, Júlio Cesar Fabris. **ASP.NET: Uma Nova Revolução na Criação de Sites e Aplicações Web**. Axcel Books, 2001. Disponível em <http://www.juliobattisti.com.br/artigos/ti/ncamadas.asp>. Acesso em 24 mar. 2004. [http://www.timaster.com.br/revista/colunistas/ler\\_colunas\\_emp.asp?cod=507](http://www.timaster.com.br/revista/colunistas/ler_colunas_emp.asp?cod=507). Acesso em 24 mar. 2004.
- BEZERRA, Eduardo. **Princípios de Análise e Projeto de Sistemas com UML**. Rio de Janeiro: Campus, 2002.
- BOOCH, Grady et al. **UML Guia do Usuário**. Rio de Janeiro: Campus, 2000.
- CANTÚ, Marco. **Dominando o Delphi 4 - a Bíblia**. São Paulo: Makron Books, 1998.
- CANTÚ, Marco. **Dominando o Delphi 7 - a Bíblia**. São Paulo: Makron Books, 2003.
- CÔRTEZ, Marco. **Delphi 6 DBExpress e Multicamadas**. Cuiabá: KCM, 2002.
- CRAWFORD, William. **Essential Multitier J2EE Design Patterns**. Disponível em <http://otn.oracle.com/oramag/oracle/03-jan/o13j2ee.html>. Acesso em 31 mar. 2004.
- FURLAN, José Davi. **Modelagem de Objetos através da UML**. São Paulo: Makron Books, 1998.
- JONES, Meilir Page. **Fundamentos do Desenho Orientado a Objeto com UML**. São Paulo: Makron Books, 2001.
- KALSING, André Cristiano. **Arquitetura de Aplicações Distribuídas – Multicamadas (N-Tier)**. Novo Hamburgo: FEEVALE, 2003.
- LARMAN, Craig. **Utilizando UML e Padrões**. Porto Alegre: Bookman, 2000.

- LIMA, Gleydson. **Java Remote Method Protocol**. Disponível em <[http://www.jspbrasil.com.br/jsp/tutoriais/tutorial.jsp?idTutorial=016\\_001](http://www.jspbrasil.com.br/jsp/tutoriais/tutorial.jsp?idTutorial=016_001)>. Acesso em 03 jun. 2004.
- LÖWY, Juval. **COM and .NET Component Services**. O'Reilly & Associates Inc, 2001.
- PAULI, Guinther de Bitencourt. **Multicamadas Passo a Passo** (Revista ClubeDelphi, Edição 46). Rio de Janeiro: Neofício, 2004.
- PELEGRINA, J. A. **DicWeb Dicionário de Informática**. Disponível em <<http://www.dicweb.com/>>. Acesso em 06 jun. 2004.
- PEREZ, Rogério. **Desenvolvimento de Aplicações em “nCamadas”**. Disponível em <[http://www.imasters.com.br/web/conteudo/coluna\\_com.php?codcoluna=346](http://www.imasters.com.br/web/conteudo/coluna_com.php?codcoluna=346)> <<http://www.imasters.com.br/web/conteudo/secao.php?codartigo=1756>>. Acesso em 10 jun. 2004.
- PRESSMAN, Roger S. Engenharia de Ssoftware. São Paulo: Makron Books, 1995.
- RODRIGUES, Anderson Haertel. **Sistemas Multicamadas com Delphi**. Florianópolis: VisualBooks, 2002.
- SALEMI, Joe. **Guia para Banco de Dados Cliente/Servidor**. Rio de Janeiro: IBPI PRESS, 1995.
- SANT'ANNA, Mauro. **SOAP e Web Services**. Disponível em <[http://www.linhadecodigo.com.br/artigos.asp?id\\_ac=38](http://www.linhadecodigo.com.br/artigos.asp?id_ac=38)>. Acesso em 01 jun. 2004.
- TEIXEIRA, Steve et al. **Delphi 6 Guia do Desenvolvedor**. Rio de Janeiro: Campus, 2002.
- TODD, Nick; SZOLKOWSKI, Mark.. **JavaServer Pages – O Guia do Desenvolvedor**. Rio de Janeiro: Campus, 2003.
- VASCONCELOS, A. M. L. **Introdução à Engenharia de Software e os Princípios de Qualidade**. Lavras: FAEP, 2002.
- VASCONCELOS, Fernando. **Desvendando Web Services (SOAP/XML)**. Disponível em <[http://www.linhadecodigo.com.br/artigos\\_impressao.asp?id\\_ac=3](http://www.linhadecodigo.com.br/artigos_impressao.asp?id_ac=3)>. Acesso em 13 jun. 2004.
- WILDEROM, Stella Martinez; WILDEROM, Bastiaan Pieter Marinus. **Aplicações Cliente/Servidor com Delphi 6 + Interbase 6**. São Paulo: Érica, 2002.