UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO

(Bacharelado)

PROTÓTIPO DE UM SISTEMA DE IMPORTAÇÃO PARA UMA AGÊNCIA DE TRANSPORTES INTERNACIONAIS UTILIZANDO TECNOLOGIAS ADO, COM/DCOM E ASP.

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

RODRIGO BARROSO GONÇALVES

BLUMENAU, DEZEMBRO/2000

PROTÓTIPO DE UM SISTEMA DE IMPORTAÇÃO PARA UMA AGÊNCIA DE TRANSPORTES INTERNACIONAIS UTILIZANDO TECNOLOGIAS ADO, COM/DCOM E ASP.

RODRIGO BARROSO GONÇALVES

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

BACHAREL EM CIÊNCIAS DA COMPUTAÇÃO

_____

Prof. Maurício Capobianco Lopes — Orientador na FURB

_____

Sra. Janaína Kretzer — Orientadora da empresa

_____

Prof. José Roque Voltolini da Silva — Coordenador do TCC

BANCA EXAMINADORA

_____

Prof. Maurício Capobianco Lopes

_____

Prof. Everaldo Arthur Grahl

_____

Prof. Ricardo Guilherme Raduns

A Deus, aos meus pais e toda minha
família, pela força que sempre me deram.

AGRADECIMENTOS

Agradeço, principalmente, aos meus pais, Florisvaldo Gonçalves da Silva e Suelly Barroso Gonçalves, que sempre batalharam para o crescimento dos filhos, mesmo nas horas mais difíceis. Sem a ajuda deles, nada disso estaria acontecendo.

Ao meu orientador, Professor Maurício Capobianco Lopes, pelo apoio e amizade nesse período acadêmico, principalmente nessa fase final.

Aos meus avós Manoel Leal Barroso, Maria do Espirito Santo Barroso Valadares e Iolanda Divina da Silva, e aos meus irmãos que sempre estiveram me apoiando e incentivando.

A Deus, por ter me guiado em todos esses anos de vida.

Aos diretores da Blucargo Transportes e a todos os colegas de trabalho, pelo ótimo ambiente profissional, em especial àqueles que contribuíram para o desenvolvimento desse trabalho.

E a todos os outros amigos e parentes que incentivaram o meu estudo.

Sumário

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS

ADO	– *ActiveX Data Objects*
API	– *Application Program Interface*
ARPA	– *Advanced Research Projects Agency*
ASP	– *Active Server Page*
AWB	– *Air Way Bill*
BDE	– *Borland DataBase Enginee*
COM	– *Component Object Model*
CCI	– *Câmara de Comércio Internacional*
CLSID	– *Class Identifier*
CORBA	– *Common Object Request Broker Architecture*
DCE	– *Distributed Computing Environment*
DLL	– *Dymamic Link Libraries*
DBMS	– *Database Manager System*
DCOM	– *Distributed Component Object Model*
FTP	– *File Transfer Protocol*
HTML	– *Hypertext Markup Language*
HTTP	– *Hypertext Transfer Protocol*
IATA	– *International Air Transport Association*
IIS	– *Internet Information Server*
LPC	– *Local Procedure Calls*
MIDAS	– *Middle-tier Application Services*
MTS	– *Microsoft Transaction Server*
ODBC	– *Open Database Engine*

OLE DB – *Object Linking and Embedding - data base*

OMC – *Organização Mundial do Comércio*

OMG – *Object Management Group*

ONU – *Organização das Nações Unidas*

ORB – *Object Request Broker*

ORPC – *Object Remote Procedure Calls*

PWM – *Personal Web Manager*

PPTP – *Point to Point Tunneling Protocol*

PC – *Personal Computer*

RPC – *Remote Procedure Calls*

SCM – *Service Control Manager*

SECEX – *Secretaria de Comércio Exterior*

SGBD – *Sistema Gerenciador de Banco de Dados*

SISCOMEX – *Sistema de Comércio Exterior*

SQL – *Structured Query Language*

TCP/IP – *Transmission Control Protocol/Internet Protocol*

UML – *Unified Modeling Language*

URL – *Uniform Resource Locator*

VMT – *Virtual Method Table*

WWW – *world wide web*

RESUMO

O propósito desse trabalho é desenvolver um protótipo de sistema de importação de cargas, demonstrando a utilização da tecnologia *ActiveX Data Objects* (ADO), COM/DCOM e ASP, ressaltando características de sua utilização na implementação de sistemas e manipulação de dados com objetos distribuídos. Para construção deste protótipo foi utilizado a ferramenta visual de programação Delphi 5.0.

ABSTRACT

The purpose of this work is to develop a prototype of a loading import system, demonstrating the use of the ActiveX Data Objects (ADO), COM/DCOM and ASP technology, pointing out characteristics of its use in systems implementation and manipulation of informations with distributed objects. The visual programming tool used to build up this prototype was Delphi 5.0.

1 INTRODUÇÃO

O quadro referencial econômico e político internacional faz com que as nações tendam à globalização de mercados industriais, de serviços e consumo. Assim, aquela atitude de ficar preso ao mesmo sistema por muitos anos não funciona mais. A globalização está presente na realidade e no pensamento, desafiando um grande número de pessoas em todo o mundo, conseguindo aprofundar ainda mais a multiplicidade de vínculos econômicos.

O crescimento das operações de comércio exterior brasileiro vem superando as expectativas e cada vez mais, além de ser obrigatório conhecer todos os parâmetros e passagens desse setor, é necessário o uso de ferramentas dinâmicas, seguras e principalmente com uma interface amigável e com acesso distribuído. Quando se fala em comércio exterior, refere-se principalmente à importação e exportação.

Importação vem a ser a remessa de um país para outro, de bens e serviços. A preparação da empresa para atividades de importação envolve uma gama variada de providências, capazes de permitir-lhe penetrar no comércio de importação e mais importante, nele permanecer ([RAT1987]). Um pensamento que a empresa pretendente a participar do comércio de importação deve ter é que qualquer deslize poderá muitas vezes, comprometer não só o ganho da operação, mas o próprio valor da compra. Para garantir a segurança das informações do processo de importação e reduzir o tempo perdido com a elaboração de documentos burocráticos, foi desenvolvido um protótipo de um sistema de importação para a empresa Blucargo Transportes Nacionais e Internacionais Ltda.

1.1 OBJETIVOS

O objetivo principal do trabalho é a especificação e implementação de um protótipo de um sistema de importação para a empresa Blucargo Transportes utilizando o ambiente Delphi e a tecnologia ADO (*ActiveX Data Objects).*

Os objetivos secundários do trabalho são:

a) demonstrar a tecnologia ADO *(Activex Data Objects)*;
b) demonstrar a tecnologia COM *(Component Object Model)* e DCOM *(Distributed Component Object Model)*;
c) utilizar a tecnologia ASP *(Active Server Pages)* para acesso ao sistema via *web*.

1.2 ORGANIZAÇÃO DO TRABALHO

O trabalho está organizado da seguinte maneira :

O primeiro capítulo apresenta a contextualização e justificativa para o desenvolvimento da proposta do trabalho e apresenta a empresa Blucargo Transportes.

O segundo capítulo apresenta uma visão geral de comércio nacional e internacional, destacando a sistemática da área de importação aplicada ao trabalho.

O terceiro capítulo aborda os temas Internet e *Active Server Pages* (ASP), para o acesso aos dados do protótipo via *web* e ainda a tecnologia ADO.

O quarto capítulo descreve uma introdução de objetos distribuídos, a especificação das tecnologias COM/DCOM, para comunicação entre as camadas do protótipo, bem como detalhes do seu funcionamento.

O quinto capítulo aborda a linguagem de modelagem de dados UML e a especificação e implementação do protótipo do sistema de importação.

O sexto capítulo apresenta as considerações finais, abrangendo as conclusões do desenvolvimento deste protótipo, bem como as dificuldades, e sugestões para novos trabalhos.

1.3 A EMPRESA

A empresa Blucargo Transportes Nacionais e Internacionais Ltda, nasceu no estado de Santa Catarina, na cidade de Blumenau em 1983. O início da suas atividades foi direcionado para o agenciamento do transporte aéreo internacional de cargas, onde alcançou grande sucesso com extrema rapidez. A empresa está classificada como sendo uma agência de cargas, ou seja, uma intermediária entre empresas exportadoras e/ou importadoras e as companhias aéreas, disponibilizando de agentes em todo o mundo, com a anuência de empresas aéreas ou marítimas, a receber embarques, emitir AWB *(Air Way Bill)* e coletar taxas e fretes inerentes ao transporte. A empresa é credenciada ao IATA *(International Air Transport Association)*, uma associação voluntária, não exclusiva, apolítica e democrática que congrega as companhias de aviação a nível mundial, cuja sede localiza-se no Canadá. Por ser uma empresa credenciada ao IATA, devem ser seguidas normas e regras de padrões internacionais.

A conquista do mercado obrigou a empresa a ampliar suas atividades, sendo que atualmente conta com uma forte estrutura em todos os segmentos relacionados ao transporte

de mercadorias no comércio internacional. Com seriedade e eficiência, foram também criadas as empresas de apoio a organização :

a) Blucargo Transporte Rodoviário de Carga;

b) Blucargo Comissária de Despachos Aduaneiros.

A Blucargo possui atendimento total aos seus clientes desde a coleta da carga até a entrega no seu destino final. Conta com uma vasta rede de agentes no Brasil. No exterior está associada a grupos fortes e agentes de transporte internacional experientes. Além disso a empresa recebeu as condecorações das companhias aéreas, como o agente de carga internacional destaque no ano de 1987.

Quanto à estrutura computacional, a empresa possui 21 computadores PC’s de última geração, e um servidor com Windows NT SERVER, todos com acesso à Internet e aos respectivos sistemas produzidos pelo departamento de informática da empresa. A empresa preza por equipamentos de qualidade, garantindo estabilidade e eficiência na sua rede de computadores. Todos os departamentos estão informatizados.

O principal motivo para o desenvolvimento do protótipo em questão, é principalmente, facilitar e dar integridade em todo o processo de agenciamento de cargas no departamento de importação, desde as entradas de informações no sistema até as saídas geradas. Outro fator importante é a diminuição de erros no sistema além da rapidez de acesso às informações e segurança dos dados, e ainda disponibilizar o acesso aos dados via *web.*

2 COMÉRCIO EXTERIOR

O comércio exterior já faz parte do cotidiano, através dos produtos estrangeiros que estão na economia : nas fábricas (máquinas e equipamentos), nos supermercados (produtos alimentícios), nas lojas (vestuário e eletrodomésticos), nas ruas (carros), oriundos de diversas regiões do planeta ([SOS2000]).

2.1 COMÉRCIO NACIONAL E COMÉRCIO INTERNACIONAL

Da perspectiva da ciência econômica, o comércio tem uma função distributiva, correspondendo a ação de buscar produtos e encaminhá-los aos consumidores. Essa função do comércio é perceptível tanto nas transações realizadas entre pessoas de um mesmo país (comércio nacional) quanto naquele praticado entre pessoas de distintas nacionalidades (comércio internacional) ([MAR1999C]).

Há uma distinção notável entre as formas de praticar comércio, que consiste no fato de estarem os sujeitos da relação de compra e venda situados em diferentes países.

Diferentemente do comércio praticado dentro dos limites de uma nação, sujeito ao respectivo ordenamento jurídico interno, o comércio internacional se pratica à luz de uma ordem de direito muito mais complexa. É que os intervenientes (o vendedor e comprador) não apenas devem estrita obediência às leis internas de seus países, como o próprio ato comercial se realiza dentro de um parâmetro legal ditado pelo ordenamento jurídico internacional, devendo existir, ainda, um necessário acoplamento entre essas duas vertentes ([MAR1999C]).

Pode-se dizer que o comércio internacional se pratica entre nações, através de indivíduos. Por exemplo, Brasil e Canadá comerciam entre si, comprando ou vendendo mercadorias. Do ponto de vista internacional, os pólos da relação comercial serão os países intervenientes, de forma que se o Canadá importar café do Brasil e o Brasil importar máquinas do Canadá, pagarão um ao outro as respectivas aquisições. A concretização dessas operações se dá entre pessoas, usualmente por compra e venda. Nesse aspecto, reside outra fundamental diferença entre o comércio interno e o externo.

No comércio interno, uma mercadoria é traditada do vendedor pelo comprador de forma direta, sendo correto dizer que o Estado, embora regule a forma como se dará essa

transferência de propriedade, não interfere, salva exceções, no tocante à posse ou propriedade do produto comercializado.

No comércio internacional, porém, a própria operação de compra e venda internacional sujeita-se a uma autorização dos Estados que jurisdicionam os agentes econômicos da transação comercial. Significa dizer que uma mercadoria não será exportada, nem importada, sem dita autorização estatal, tal como disposto nas respectivas legislações, de modo que o aperfeiçoamento do contrato de compra e venda internacional condiciona-se à vontade soberana dos Estados envolvidos, segundo as conveniências da política comercial que adotam ([MAR1999C]).

Portanto, comércio nacional e internacional são representativos de uma atividade mercantil, de distribuição e circulação de riquezas. Porém, o comércio internacional é dotado de uma maior complexidade quando comparado à mesma atividade no plano interno (nacional).

2.2 COMÉRCIO INTERNACIONAL E COMÉRCIO EXTERIOR

O conjunto normativo que define uma operação de exportação ou importação divide-se em normas de comércio exterior do Brasil e normas de comércio internacional.

Dentro desse prisma, deve-se separar as questões internacionais, que são os estudos e as operações de trocas entre países distintos, caracterizando-se pelo intercâmbio de mercadorias, serviços e movimentação de capitais, que chama-se comércio internacional, e dos termos, regras e normas nacionais das transações e estudos realizados no comércio internacional, que chama-se comércio exterior ([SOS2000]).

As normas de comércio internacional são aquelas que se aplicam, uniformemente, a mais de um país, visando a facilitação dos negócios internacionais. Como área específica, o comércio internacional é percebido nas trocas comerciais havidas entre as diversas nações que compõem a comunidade mundial. As regras são disciplinadas através de acordos entre dois ou mais países ou criadas a partir de organismos internacionais como a Organização das Nações Unidas (ONU), Câmara Internacional de Comércio (CCI) ou Organização Mundial do Comércio (OMC) ([MAR1999C]).

As normas de comércio exterior do Brasil são aquelas emanadas dos órgãos do executivo federal, que disciplinam a entrada no país de mercadorias procedentes do exterior e

a saída de mercadorias do território nacional, com suas repercussões nas áreas tributária, administrativa, comercial, aduaneira e financeira.

A realização de qualquer negócio de importação ou exportação resultará sempre no estudo de três conjuntos normativos: comércio internacional, comércio exterior do Brasil e o comércio exterior do país com o qual o Brasil estará negociando.

2.3 IMPORTAÇÃO

As importações têm sido alvo de interesse das pequenas, médias e grandes empresas como solução alternativa e inteligente para a expansão de sua área comercial de produção. O mesmo acontece com pessoas físicas, que vislumbram, na importação, uma saída para ampliar ou efetivar seus projetos.

As empresas interessadas em efetuar importações, inscrevem-se no Registro de Exportadores e Importadores da Secretaria de Comércio Exterior (SECEX). Quanto às pessoas físicas é reservado o mesmo direito, uma vez que as operações não revelem prática de comércio.

Realizado o contato e definidos os produtos e as condições da operação, o importador deverá solicitar ao exportador estrangeiro a remessa de um documento que formalize o preço praticado na operação (cartas, telex, fax, telegramas, ordens de compra ou contratos), porque a qualquer época a SECEX poderá solicitar do importador informações ou documentação pertinente.

O licenciamento das importações é efetuado por meio do SISCOMEX (sistema de comércio exterior). Para se obter a senha que permite ao usuário o acesso ao SISCOMEX faz-se necessário apenas o seu credenciamento junto à Secretaria da Receita Federal.

Concretizada a operação comercial, o importador poderá autorizar o embarque da mercadoria ao exterior, ressaltando, que as mercadorias e/ou operação sujeitas a anuência prévia de importação exigirão o cumprimento antecipado desta condição.

Após o embarque, o exportador remeterá, de acordo com a modalidade de pagamento convencionada, os documentos que permitirão ao importador liberar as mercadorias na alfândega brasileira. Dentre esses documentos destacam-se :

a) conhecimento de embarque;
b) fatura comercial;

c) certificado de origem (quando o produto for objeto de acordos internacionais);

Com relação ao pagamento da mercadoria ao exportador, existem algumas maneiras de efetuá-lo :

a) antecipado (pagamento da mercadoria antes do embarque da mesma);
b) cobrança (pagamento após o recebimento da mercadoria);
c) carta de crédito (emitida pelo banco, obedecendo certas cláusulas impostas pelo exportador).

As operações de compra e venda de moedas estrangeiras, realizadas entre uma pessoa ou empresa e um estabelecimento autorizado a operar em câmbio, são formalizadas através de um contrato de câmbio, conforme modelo próprio e de acordo com as normas estabelecidas pelo Banco Central do Brasil. Com a chegada da mercadoria no Brasil, inicia-se a fase de liberação na alfândega brasileira.

O importador ou Despachante Aduaneiro, com base na documentação correspondente à Liberação de Importação (LI), conhecimento de embarque e fatura comercial – elaborará a Declaração de Importação (DI) no SISCOMEX e, mediante o pagamento do Imposto de Importação e do IPI (Imposto sobre Produtos Industrializados) comprovados pelo Documento de Arrecadação de Receitas Federais (DARF), efetuará o seu registro no mesmo sistema.

Juntamente com todos esses documentos e o comprovante de recolhimento ou exoneração do ICMS, deverá o importador apresentar o extrato da Declaração de Importação à Receita Federal do local onde estiver a mercadoria para conclusão do denominado Despacho Aduaneiro.

O Despacho Aduaneiro é um conjunto de atos praticados pelo Fiscal da Receita Federal, que tem por finalidade o desembaraço aduaneiro (autorização da entrega da mercadoria ao importador) mediante a conclusão da conferência da mercadoria, o cumprimento da legislação tributária e a identificação do importador. No SISCOMEX a Receita Federal emitirá um Comprovante de Importação (CI) que comprovará a liberação alfandegária.

A retificação de informações prestadas na DI, a alteração de cálculos e a indicação de multas e acréscimos legais serão feitas através de procedimento específico no SISCOMEX.

3 INTERNET E ASP

Atualmente, uma empresa seja ela pequena, média ou de grande porte que tenha relação direta ou indireta com o comércio exterior, dificilmente sobreviverá sem o uso da Internet. Como exemplo, 70% (setenta porcento) da comunicação da empresa Blucargo Transportes com seus clientes e fornecedores se dá através da Internet, através de correio eletrônico ou páginas www. Mesmo a empresa envolvida que não estiver interessada em se conectar à Internet, sofrerá uma forte pressão das partes envolvidas, pela rapidez e custo de transmissão que a Internet proporciona.

A Internet nasceu em 1969, nos Estados Unidos, com o nome de ARPAnet (ARPA: *Advanced Research Projects Agency*) interligando, originalmente, laboratórios de pesquisa e sendo uma rede do departamento de defesa norte-americano. Era o auge da Guerra Fria, e os cientistas queriam uma rede que continuasse de pé em caso de um bombardeio. Surgiu então o conceito central da Internet : é uma rede em que todos os pontos se eqüivalem e não há um comando central. Assim, se B deixa de funcionar, A e C continuam a poder se comunicar.

O nome Internet propriamente dito surgiu bem mais tarde, quando a tecnologia da ARPAnet passou a ser usada para conectar universidades e laboratórios, primeiro nos EUA e depois em outros países.

O que essas redes têm em comum é o protocolo TCP/IP *(Transmission Control Protocol/Internet Protocol)*, que permite que elas se comuniquem umas com as outras. Esse protocolo é a língua comum dos computadores que integram a Internet.

A Internet possui vários serviços distintos disponíveis, sendo que os principais são :

a) correio eletrônico : ou *e-mail*, é um dos serviços mais elementares e mais importantes disponíveis na Internet. Basicamente, o correio eletrônico é a troca de mensagens que o usuário da Internet pode mandar para outro usuário;

b) www *(world wide web)* : a teia mundial está em crescimento explosivo. Tem registrado recordes de crescimento por volumes de dados transmitidos por mês e tem sido responsável pelo aumento da capacidade de tráfego em muitos canais de comunicação. A w*ww* é uma rede virtual (não-física) "sobre" a Internet, que torna os serviços disponíveis na Internet, totalmente transparentes para o usuário e ainda possibilita a manipulação multimídia da informação. Assim, qualquer usuário

pode, somente usando o mouse, ter acesso a uma quantidade enorme de informações na forma de imagens, textos, sons, gráficos, vídeos e etc, navegando através de palavras-chaves e ícones. As informações são disponibilizadas através de páginas HTML *(HyperText Markup Language);*

c) FTP (*File Transfer Protocol)* : é o protocolo usado na Internet para transferência de arquivos entre computadores. Basicamente os programas que implementam o FTP fazem transferência de arquivos entre seu computador local e outro remoto.

Com o decorrer dos anos, muita tecnologia foi acoplada à Internet, com o objetivo de interagir cada vez mais a rede com o dia a dia das pessoas. Neste trabalho, por exemplo, foi utilizada a tecnologia *Active Server Pages* (ASP), uma linguagem para geração de páginas HTML dinamicamente, pelo fato de o protótipo de sistema de importação disponibilizar acesso pela Internet. ASP é uma linguagem intermediária que possibilita a comunicação entre os dados de sistemas locais com uma página da Internet. Uma página ASP é uma página HTML que possui programas acoplados que são processados no servidor *web* antes da página ser enviada ao usuário.

Segundo [MAR1999A], sempre que uma página ASP é requisitada pelo *browser*, esta página é executada pelo servidor, e a partir daí é gerada uma página HTML, a qual é enviada para o *browser* que a requisitou. A linguagem ASP não serve apenas para consultas a banco de dados, serve também para envio e recebimento de correio eletrônico, via páginas HTML, criação de rotinas de propaganda rotativa, identificação e autenticação de usuários, leitura de arquivos texto e uma infinidade de outras aplicações.

3.1 OBJETOS ASP

O ASP é composto de cinco objetos internos. Trata-se de estruturas especiais que possuem propriedades, métodos, eventos, coleções e etc ([MIT2000]). Dentre outras coisas, tais objetos servem para :

a) verificar dados informados pelo clientes *web*;
b) enviar respostas HTML para tais clientes;
c) instanciar objetos *ActiveX* em seus *scripts*;
d) permitir a comunicação entre clientes conectados ao aplicativo ASP.

Os objetos internos do ASP estão na tabela 1.

TABELA 1 - OBJETOS INTERNOS DO ASP

Application	representa um conjunto de páginas de um mesmo diretório virtual
Session	representa uma sessão aberta com um cliente via *web browser*
Server	representa o servidor *web* em si, permitindo acesso a algumas propriedades do mesmo e a criação de instâncias de objetos *Activex*
Response	representa as respostas HTML enviadas ao cliente
Request	representa os dados enviados por um formulário HTML ao servidor *web*

3.1.1 APPLICATION

Ao conjunto de páginas ASP de um mesmo diretório virtual dá-se o nome de aplicação ASP. Tal aplicação será iniciada na primeira vez que um usuário tentar acessar alguma página desse diretório virtual e será finalizada quando o servidor *web* for desligado ([MAR1999A]).

O objeto *Application* existe para possibilitar o armazenamento e recuperação de valores relacionadas a uma aplicação ASP. Com ele pode-se criar variáveis de qualquer subtipo cujo valor pode ser acessado ou modificado por qualquer usuário conectado ao diretório virtual.

Para criar uma variável do nível de aplicação, deve-se escrever comandos com a seguinte sintaxe :

Application(“NOME_DA_VARIAVEL”) = VALOR_DA_VARIAVEL

Uma vez criada, tal variável estará acessível a qualquer usuário da aplicação. Seu valor ficará armazenado até que o servidor *web* seja desligado.

Como o conteúdo desse tipo de variável pode ser modificado por qualquer usuário conectado à aplicação, poderia haver alguma confusão se vários usuários tentassem alterar esse valor ao mesmo tempo. Para evitar possíveis problemas de concorrência, o objeto *Application* disponibiliza dois métodos : LOCK e UNLOCK ([MIT2000]).

O primeiro bloqueia as variáveis de nível de aplicação para o usuário que invoca tal método. Se qualquer outro “usuário” tentar acessar variáveis desse nível, ficará esperando até a aplicação ser desbloqueada.

A aplicação só será desbloqueada quando o *script* (pequenos programas embutidos) que a bloqueou termina sua execução, ou quando ocorre o “*TimeOut*” (tempo de espera), ou quando o *script* invoca o método UNLOCK.

Ainda relacionado a esse objeto existem dois eventos apresentados na tabela 2.

TABELA 2 - EVENTOS DO OBJETO APPLICATION

Application_OnStart	ocorre quando a aplicação é iniciada, ou seja, quando um diretório virtual é acessado pela primeira vez
Application_OnEnd	ocorre quando a aplicação é finalizada, ou seja, quando o *web server* é desligado

Um evento é uma subrotina automaticamente chamada quando o sistema sofre alguma ação especifica. Tais subrotinas não são escritas diretamente nas páginas ASP mas num arquivo a parte nomeado de GLOBAL.ASA.

Sendo assim, quando um diretório virtual for acessado pela primeira vez, o servidor *web* procura em tal diretório a existência desse arquivo. Se encontra, abre o arquivo e procura a subrotina *Application_OnStart* para executar seus comandos. A mesma coisa acontece quando se desliga o servidor *web*, só que ele chama a subrotina *Application_OnEnd*.

No quadro 1 apresenta-se um exemplo onde foi criada uma variável de nível de aplicação chamada “DataHoraI” para armazenar a data e hora em que a aplicação foi iniciada. Outra variável chamada “Título” para armazenar o título da aplicação ASP, e uma variável chamada “Correio” que armazena o *e-mail* (endereço eletrônico) do w*ebmaster* (programador de aplicações Internet).

QUADRO 1 - EXEMPLO ASP DE DATA E HORA

GLOBAL.ASA

```
<SCRIPT LANGUAGE=VBSCRIPT RUNAT=SERVER>
Sub Application_OnStart()
   Application(“DataHoraI”)=Now
  Application(“Titulo”)=”EXEMPLO DE ASP”
  Application(“Correio”)=”mailto:cpd.bluc@zaz.com.br”
End Sub
</SCRIPT>
```

Exemplo : APPLICATION1.ASP

```
<% @LANGUAGE=VBSCRIPT %>
<HTML><HEAD><TITLE><%=Application(“Titulo”)%></TITLE></HEAD>
<BODY>
Essa aplicação foi iniciada em <B><%=Application(“DataHoraI”)%></B><BR>
```

```
<A Href="<%=Application("Correio")%>">Web Master</a>
</BODY>
</HTML>
```

É importante destacar que só poderá existir um arquivo GLOBAL.ASA em cada diretório virtual.

Deve-se observar que o trabalho de manutenção do site pode ficar facilitado. Se todas as páginas ASP do diretório virtual possuírem um padrão de cores, links, cabeçalhos ou etc, as páginas teriam muito código em comum, então, se forem necessárias modificações, deveriam ser alterados todos os arquivos do diretório virtual. Mas se forem utilizadas variáveis de nível de aplicação para armazenar essas configurações, não seria necessário mudanças em todos os arquivos do diretório, mas só no arquivo GLOBAL.ASA. Este exemplo é mostrado no quadro 2.

QUADRO 2 - EXEMPLO GLOBAL.ASA

```
<SCRIPT LANGUAGE=VBSCRIPT RUNAT=SERVER>
Sub Application_OnStart()
  Application(“DataHoraI”)=Now
  Application(“Titulo”)=”Exemplo de ASP by Rodrigo”
  Application(“Correio”)=mailto:cpd.bluc@zaz.com.br
 Application(“CorFundo”)=”Black”
  Application(“CorTexto”)=”Yellow”
  Application(“TamFonte”)=”4”
End Sub
</SCRIPT>
```

Exemplo : APPLICATION2.ASP

```
<% @LANGUAGE=VBSCRIPT %>
<HTML><HEAD><TITLE>
<%=Application("Titulo")%>
</TITLE></HEAD>
<BASEFONT SIZE=<%=Application("TamFonte")%>
<COLOR=<%=Application("CorTexto")%>
<BODY BGCOLOR=<%=Application("CorFundo")%>>
```

```
        Essa aplicação ASP foi iniciada em
<B><%=Application("DataHoraI")%></B><BR>
        <A Href="<%=Application("Correio")%>">Web Master</a>
        </BODY>
        </HTML>
```

3.1.2 SESSION

Toda vez que um usuário *web* se conecta a um aplicativo ASP é iniciada uma sessão para o mesmo no servidor *web*. Para representar tal sessão, o ASP possui um objeto interno chamado *Session* ([MIT2000]).

Na verdade, ele é muito parecido com o objeto *Application*. A diferença está em dizer que esse objeto pode armazenar valores ligados apenas a um único visitante do site (o dono da sessão). Com ele pode-se criar variáveis de qualquer subtipo cujo valor pode ser acessado ou modificado somente pelo "dono" da sessão.

Para criar uma variável do nível de sessão, deve-se escrever comandos com a seguinte sintaxe :

Session("NOME_DA_VARIAVEL") = VALOR_DA_VARIAVEL

As variáveis de sessão permanecerão na memória (ativas) até que a sessão seja encerrada. Isso pode acontecer quando o usuário fechar o *web* browser, quando ocorre o *"TimeOut"* da sessão, ou quando o *script* invoca o método ABANDON do objeto *Session* ([MIT2000]).

A propriedade *"TimeOut"* é usada quando o usuário fica parado "sem fazer nada" no *browser*. O tempo *default* (propriedade padrão) é de vinte minutos, mas esse valor pode ser modificado da seguinte forma :

Session.Timeout = VALOR_MINUTOS

Ainda relacionado a esse objeto existem dois eventos como apresentado na tabela 3.

TABELA 3 - EVENTOS DO OBJETO SESSION

Session_OnStart	ocorre quando a sessão é iniciada
Session_OnEnd	ocorre quando a sessão é finalizada.

À exemplo dos eventos do objeto *Application*, eles também devem ser escritos como subrotinas de um arquivo GLOBAL.ASA.

O exemplo do quadro 3 ilustra o conceito de sessão. Tem-se uma variável a nível de aplicação chamada “contador”. Ela serve para informar a quantidade de pessoas que acessaram essa aplicação ASP. A idéia é incrementar o valor dessa variável toda vez que uma sessão é iniciada. Também utiliza-se uma variável de sessão que informa a hora em que a sessão em questão foi aberta.

QUADRO 3 - GLOBAL.ASA EXEMPLIFICANDO O OBJETO SESSION

```
<SCRIPT LANGUAGE=VBSCRIPT RUNAT=SERVER>
Sub Application_OnStart()
  Application(“DataHoraI”)=Now
  Application(“Titulo”)=”Exemplo de ASP by Rodrigo”
  Application(“Correio”)=”mailto:cpd.bluc@zaz.com.br”
  Application(“CorFundo”)=”Black”
  Application(“CorTexto”)=”Yellow”
  Application(“TamFonte”)=”4”
  Application(“Contador”)=0
End Sub
Sub Session_OnStart()
    Application("Contador")=Application("Contador")+1
   Session(“HoraS”)=Time
End Sub
</SCRIPT>
```

Exemplo : SESSAO1.ASP

```
<% @LANGUAGE=VBSCRIPT %>
<HTML><HEAD><TITLE>
</TITLE></HEAD>
Essa sessão foi iniciada às <%=Session(“HoraS”)%><BR>
Você é o visitante de número <%=Session(“Contador”)%><BR>
Desde <%=Application(“DataHoraI”)%>
<HR>
```

```
<A HREF=”sessao2.asp”>Encerrar Sessão</A>
</BODY>
</HTML>
```

3.1.3 RESPONSE

Antes de uma página ASP ser enviada ao cliente, o *webserver* (servidor de páginas Internet) lê seu conteúdo. Se encontra *TAGS HTML* (blocos de programas HTML) ou texto, envia diretamente ao cliente. Se encontra *scripts*, executa e repassa-os o resultado HTML para o *browser*. Para representar essas respostas HTML, ASP possui o objeto interno *Response* ([MAR1999A]).

Com esse objeto pode-se enviar simples comandos HTML ou texto, redirecionar o *browser* para buscar informações em outro URL *(Uniform Resource Locator)*, *bufferizar* (armazenar em memória temporária) a resposta ao cliente, bem como interromper o envio da página ASP.

Para tanto, esse objeto possui as propriedades e métodos como na tabela 4.

TABELA 4 - PROPRIEDADES E MÉTODOS DO OBJETO RESPONSE

Write Texto	com esse método pode-se enviar qualquer texto ao *browser*
End	método que termina o envio da página ASP ao cliente, mesmo que ela não tenha chagado ao final
Buffer = Valor Booleano	quando *Buffer=True*, o servidor *web* só enviará a página ASP ao cliente quando encerrar toda a leitura da mesma, ou quando for utilizado nesse *script* o metódo *Flush*. Essa propriedade só pode ser utilizada antes de qualquer comando que gere respostas HTML ao cliente
Redirect URL	permite redirecionar o *browser* do cliente para outra página(URL). Se a propriedade *Buffer* for diferente de *True*, esse método só poderá ser chamado antes de qualquer comando que gere respostas HTML ao cliente
Clear	esse método apaga todo o conteúdo do *Buffer* se o mesmo estiver ativo
Flush	esse método pode ser utilizado para enviar o conteúdo do *Buffer* para o cliente *web*

Cookies("nome_cookie") = valor	cria *cookies*[1] ou altera seu valor. Se a propriedade *Buffer* for diferente de *True*, essa propriedade só poderá ser chamado antes de qualquer comando que gere respostas HTML ao cliente
Expires = minutos	essa propriedade informa o tempo (minutos) em que uma página ASP pode permanecer ativa na cache do *web browser*
ExpiresAbsolute = #data hora#	essa propriedade informa a data e a hora exata em que a página expira da cache do *web browser*
AppendToLog Texto	esse método escreve um texto no arquivo de LOG do *web server*

Com o objeto *Response* pode ser escrita uma página ASP 100% *script*, como apresentado no exemplo do quadro 4.

QUADRO 4 - EXEMPLO DO OBJETO RESPONSE

RESPONSE1.ASP

```
<SCRIPT LANGUAGE=VBSCRIPT RUNAT=SERVER>
RESPONSE.WRITE "<HTML><HEAD><TITLE>Exemplo de ASP</TITLE></HEAD>"
RESPONSE.WRITE "<BODY><CENTER>Testando objeto Response</CENTER>"
RESPONSE.WRITE "</BODY></HTML>"
</SCRIPT>
```

Pode-se também escrever uma página ASP que simplesmente redireciona o usuário para outro site, como apresentado no quadro 5.

QUADRO 5 - REDIRECIONAMENTO NO OBJETO RESPONSE

Exemplo : RESPONSE2.ASP

```
<% @LANGUAGE=VBSCRIPT %>
<% RESPONSE.REDIRECT "http://www.blucargo.com.br"%>
```

Quando utiliza-se *Buffer=True*, tem-se mais controle do que será enviado como resposta ao cliente, como apresentado no quadro 6.

[1] *Cookies* são pequenos trechos de informações, como *strings* (caracteres) e valores numéricos, armazenados no computador do cliente por uma quantidade especificada de tempo ([MIT2000]).

QUADRO 6 - EXEMPLO DO OBJETO RESPONSE

Exemplo : RESPONSE3.ASP

```
<% @LANGUAGE=VBSCRIPT %>
<% RESPONSE.BUFFER=TRUE
 DIM Inicio, Fim
Inicio="<HTML><HEAD><TITLE>Exemplo ASP</TITLE></HEAD><BODY>"
 Fim = "</BODY></HTML>"
RESPONSE.WRITE Inicio
RESPONSE.WRITE "Esse texto não será enviado ao Cliente!"
RESPONSE.WRITE Fim
RESPONSE.CLEAR
RESPONSE.WRITE Inicio & Chr(13)
RESPONSE.WRITE "Esse texto será enviado ao cliente!" & Chr(13)
RESPONSE.WRITE Fim %>
```

3.1.4 REQUEST

Esse objeto serve para possibilitar a captura em páginas ASP dos dados passados pelos formulários HTML ao servidor *web* ([MIT2000]).

Existem três formas diferentes de fazer essa captura, dependendo do METHOD (Tabela 5).

TABELA 5 - MÉTODOS DO OBJETO REQUEST

Get	Request.QueryString(Nome_Objeto)
Post	Request.Form(Nome_Objeto)
Get ou Post	Request(Nome_Objeto)

O quadro 7 mostra um exemplo do objeto *Request*, controlando o acesso à informações de um site.

QUADRO 7 - EXEMPLO DO OBJETO REQUEST

REQUEST1.ASP

```
<% @ LANGUAGE= VBSCRIPT %>
<HTML><HEAD><TITLE>Exemplo de ASP</TITLE></HEAD>
<BODY>
<FORM ACTION="REQUEST2.ASP" METHOD=GET>
Usuário <INPUT TYPE=TEXT NAME="user" MAXLENGTH=10><BR>
Senha <INPUT TYPE=PASSWORD NAME="senha" MAXLENGTH=10><BR>
<INPUT TYPE=SUBMIT VALUE=”Enviar”>
<INPUT TYPE=RESET VALUE=”Limpar”>
</FORM></BODY>
</HTML>
```

REQUEST2.ASP

```
<% @ LANGUAGE=VBScript %>
<HTML><HEAD><TITLE>Exemplo de ASP</TITLE></HEAD>
<BODY>
Usuário = <%=Request.QueryString(“user”)%><BR>
Senha = <%=Request(“senha”)%>
</BODY>
</HTML>
```

Deve-se observar que o método GET não é adequado para formulários que exigem alguma informação secreta (senha) pois o que foi digitado no mesmo é mostrado na caixa de endereço do *browser*.

Com o objeto *Request* é possível também executar as operações descritas na tabela 6.

TABELA 6 - OUTRAS FUNÇÕES DO OBJETO REQUEST

Ler Cookies	Request.Cookies(“NomeCookie”)
Saber o endereço IP do usuário	Request.ServerVariables(“REMOTE_ADDR”)
Saber o METHOD utilizado	Request.ServerVariables(“REQUEST_METHOD”)

O quadro 8 mostra um exemplo de como saber o endereço IP *(Internet Protocol)* do usuário.

QUADRO 8 - EXEMPLO DO OBJETO REQUEST

```
REQUEST3.ASP
<% @ LANGUAGE=VBScript %>
<HTML><HEAD><TITLE>Exemplo de ASP</TITLE></HEAD>
<BODY>
Endereço IP do usuário :
<%=Request.servervariables("REMOTE_ADDR")%>
</BODY>
</HTML>
```

Como *Request.QueryString*, *Request.Form* e *Request.Cookies* possuem coleções de variáveis, as mesmas podem ser lidas com um comando *For Each*.

O quadro 9 ilustra um exemplo do objeto *Request* usando o método *Request.ServerVariables.*

QUADRO 9 - EXEMPLO DO OBJETO REQUEST

```
REQUEST4.ASP
<% @LANGUAGE=VBSCRIPT %>
<HTML><HEAD><TITLE>Exemplo de ASP</TITLE></HEAD>
<BODY>
<%FOR EACH ITEM IN Request.ServerVariables
        Response.write ITEM & " – " & Request.ServerVariables(ITEM) & "<BR>"
    NEXT%>
</BODY></HTML>
```

3.1.5 SERVER

Representa o servidor *web*, permitindo acesso à algumas de suas propriedades. Além disso, possibilita a instanciação em páginas ASP de componentes *ActiveX* escritos por outros programadores.

Os objetos *ActiveX* fazem parte da tecnologia COM para criação de componentes de software reutilizáveis. A idéia é desenvolver objetos com esse padrão para que possam ser reutilizados por "qualquer" linguagem de programação ([MIT2000]).

Em particular, o ASP pode trabalhar com componentes COM, só que os mesmos não podem possuir interface visual. Sendo assim, em páginas ASP usam-se os chamados *ActiveX DLL`s (Dymamic Link Libraries)*, componentes que não fornecem nenhuma interface com o usuário, somente com aplicativos (métodos, propriedades), como apresentado na tabela 7.

Esses componentes estendem bastante o poder de programação. Dentre outras coisas, eles permitem ([CHR1999]) :

a) acessar banco de dados;

b) criar, ler e alterar arquivos;

c) enviar *e-mail*.

TABELA 7 - PROPRIEDADES E MÉTODOS DO OBJETO SERVER

ScriptTimeOut	propriedade que determina o tempo máximo(segundos) que um *script* ASP pode ficar executando, antes que o servidor *web* o termine. Isso serve para proibir que scrits ASP fiquem executando "infinitamente"
HTMLEncode	método que codifica uma *string* para o formato HTML.
MapPath	método que retorna o PATH real de um determinado diretório virtual do servidor *web*
URLEncode	esse método transforma uma *string* para o formato padrão de URL
CreateObject	cria uma instância de um componente *ActiveX* na página ASP : *Set Obj = Server.CreateObject("IDObjeto")*

Para exemplificar o uso do objeto *Server* será usado o componente *ActiveX FileSystemObject*. Tal componente possui uma série de propriedades e métodos para manipulação de arquivos e diretórios do servidor *web*.

Para instanciar tal objeto numa página ASP, é apresentado o código do quadro 10.

QUADRO 10 - EXEMPLO DO OBJETO SERVER

```
Dim Objeto
Set Objeto = Server.CreateObject("Scripting.FileSystemObject")
```

Esse objeto recém instanciado representa o sistema de arquivo do servidor *web*. Deve-se agora criar outro objeto *(TextStream)*, a partir desse, para representar um determinado arquivo. Será utilizado o método *OpenTextFile* do *FileSystemObject* :

Set Arquivo = Objeto.OpenTextFile(Nome,modo,cria,formato)

Onde:

a) **Nome :** nome do arquivo a ser utilizado pelo *script;*

b) **Modo :** modo de abertura do arquivo, 1 para leitura, 2 para gravação por cima, 3 para gravação adicional;

c) **Cria :** valor *booleano* (lógico) que indica se o arquivo deve ser criado *(true)* ou não *(false)* caso não exista;

d) **Formato :** indica o formato de gravação do arquivo a ser utilizado, –1 *Unicode*, 0 *ASCII.*

Para ler o conteúdo de um arquivo, utilizam-se os métodos do objeto *TextStream* apresentados na tabela 8.

TABELA 8 - MÉTODOS DE LEITURA DO OBJETO TEXTSTREAM

Read (quantidade)	lê um determinado número de caracteres do arquivo
ReadLine	lê uma linha inteira do arquivo
ReadAll	lê o arquivo inteiro de uma só vez

Para gravar o conteúdo de um arquivo, utilizam-se os métodos do objeto *TextStream* apresentados na tabela 9.

TABELA 9 - MÉTODOS DE GRAVAÇÃO DO FILESYSTEMOBJECT

Write	grava uma *string* no arquivo
WriteLine	grava um *string* no arquivo, incluindo a quebra de linha
WriteBlankLines	grava um determinado número de linhas em branco num arquivo

O componente *ActiveX TextStream* ainda possui as propriedades apresentados na tabela 10.

TABELA 10 - PROPRIEDADESMÉTODOS DE GRAVAÇÃO DO OBJETO

AtEnOfLine	indica o fim de uma determinada linha do arquivo
AtEnOfStream	indica o final do Arquivo
Column	indica a coluna atual do arquivo
Line	Indica o número da linha atual do arquivo

O exemplo do quadro 11 mostra como abrir o arquivo “pessoas.txt” localizado no servidor *web* e exibir seu conteúdo.

QUADRO 11 - EXEMPLO DE COMO LER UM ARQUIVO NO SERVIDOR WEB

ARQUIVO1.ASP

```
<% @Language=vbScript %>
<HTML><HEAD><TITLE>Exemplo de ASP</TITLE></HEAD>
```

```
<BODY><CENTER>
<% dim final
  final = "</CENTER></BODY></HTML>"
  On Error Resume Next
  Set Obj = Server.CreateObject("Scripting.FileSystemObject")
  Set arquivo = Obj.OpenTextFile("D:\pessoas.txt",1)
  if Err then
     Response.write "Ocorreu um erro tentando abrir o arquivo!"
     Response.write final
     Response.End
  end if
  Response.Write "Lista de E-mails<BR>"
  Response.write "<HR>"
  do while arquivo.AtEndOfStream=false
    a = arquivo.Readline
    response.write a & "<BR>"
    a = arquivo.Readline
    response.write a & "<HR>"
  loop
  Response.Write "Nova Entrada"
  Response.Write "<FORM ACTION=Arquivo2.asp>"
  Response.Write "NOME : <INPUT TYPE=TEXT NAME=NOME><BR>"
  Response.Write "EMAIL : <INPUT TYPE=TEXT NAME=EMAIL><BR>"
  Response.Write "<INPUT TYPE=SUBMIT VALUE=ENVIAR></FORM>"
  REsponse.Write final
  arquivo.close
%>
```

A próxima página ASP (quadro 12) mostra como escrever num arquivo localizado no servidor *web*. Lembrando que essa página deve ser acessada por um formulário *web* com um objeto de *NAME=Nome* e outro *NAME=email.*

QUADRO 12 - EXEMPLO DE ESCRITA NUM ARQUIVO NO SERVIDOR WEB

ARQUIVO2.ASP

```
<% @Language=vbScript %>
<HTML><HEAD><TITLE>Exemplo de ASP</TITLE></HEAD>
<BODY><CENTER>
<% dim final
  final = "</CENTER></BODY></HTML>"
  On Error Resume Next
  Set Obj = Server.CreateObject("Scripting.FileSystemObject")
  Application.Lock
  Set arquivo = Obj.OpenTextFile("D:\pessoas.txt",8)
  if Err then
     Response.write "Ocorreu um erro tentando abrir arquivo!"
     Response.write final
     Response.End
  end if
  Arquivo.WriteLine(Request("Nome"))
  Arquivo.WriteLine(Request("Email"))
  arquivo.close
  if Err then
     Response.write "Ocorreu um erro tentando gravar no arquivo!"
     Response.write final
     Response.End
  Else
    Response.Write "Dados inseridos com sucesso!" & "<BR>"
    Response.Write "Nome:" & Request("Nome") & "<BR>"
    Response.Write "Email :" & Request("Email")
  end if
  Response.Write final
%>
```

3.2 ASP COM ACESSO À BANCO DE DADOS

É possível criar páginas ASP que tenham a habilidade de recuperar ou alterar informações em um banco de dados. Para tal, deve-se utilizar o componente *Microsoft Data Access* que já vem instalado com o IIS *(Internet Information Server)*, que é um conjunto de serviços, como FTP, HTTP, *web* e outros serviços do sistema operacional Windows NT. São programas para controlar e administrar *web sites*, e suportam aplicações *web* que têm interação com bases de dados. O uso do IIS resulta num servidor de páginas mais rápido e seguro.

Para implementar esse acesso a um banco de dados relacional o *Data Access* utiliza componentes inseridos no pacote ADO *(Activex Data Object)*, uma tecnologia baseada no ODBC *(Open DataBase Connectivity)* ([MIT2000]).

3.2.1 ODBC

O ODBC *(Open DataBase Connectivity)* é o meio mais conhecido para acessar um banco de dados em ambiente Windows. Ele facilita a vida do desenvolvedor mascarando as particularidades de implementação dos banco de dados com que os mesmos irão trabalhar.

A idéia é que cada fabricante de banco de dados crie um driver ODBC. Sendo assim, quando um programa tenta acessar determinado banco de dados via ODBC o Windows procura um driver apropriado para este, verifica como deve proceder e então efetua as operações requisitadas.

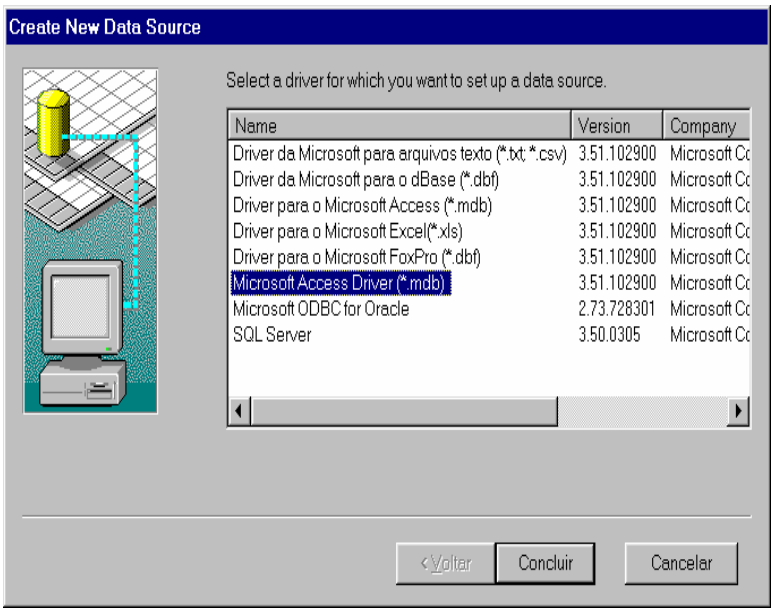
Antes de colocar a mão na massa e desenvolver páginas ASP com acesso a banco de dados deve-se :

a) criar um banco compatível com ODBC;
b) registrar o banco como uma origem de dados ODBC.

Esse último passo permite que aplicações que acessem banco de dados via ODBC precisem apenas saber o nome dessa origem de dados, dispensando-se as preocupações com a localização exata e o tipo de banco de dados acessado ([MIT2000]).

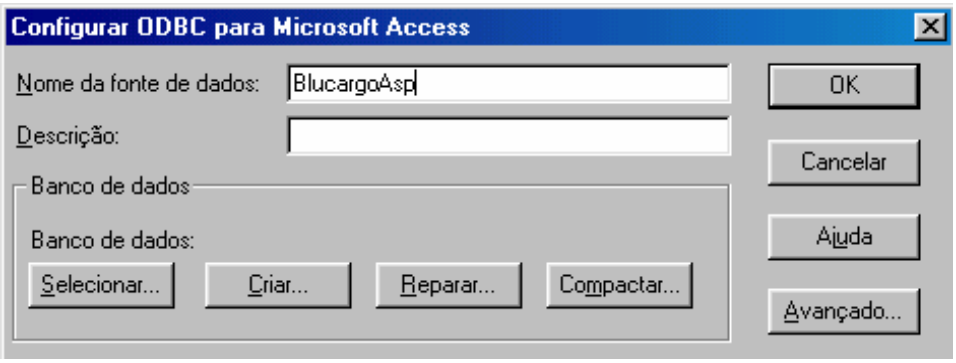
Para criar uma origem de dados deve-se executar o programa ODBC do Painel de Controle do Windows. Na janela apresentada na figura 1 deve-se escolher o driver ODBC a ser utilizado para acessar o banco de dados.

FIGURA 1 - CRIANDO UM NOVO DATASOURCE



Como exemplo, será usado o *Microsoft Access*. O nome da fonte de dados será "BlucargoAsp" (figura 2).

FIGURA 2 - PATH (NOME E CAMINHO) DA FONTE DE DADOS



Deste modo, a fonte de dados foi criada. Quando as aplicações necessitarem abrir tal banco de dados, basta informar agora o nome dessa fonte "BlucargoAsp".

3.2.2 ADO

Durante muitos anos foram desenvolvidos programas acessando bancos de dados relacionais através do ODBC. Os DBMS's *(Database Manager System)* (Oracle, dBase etc.)

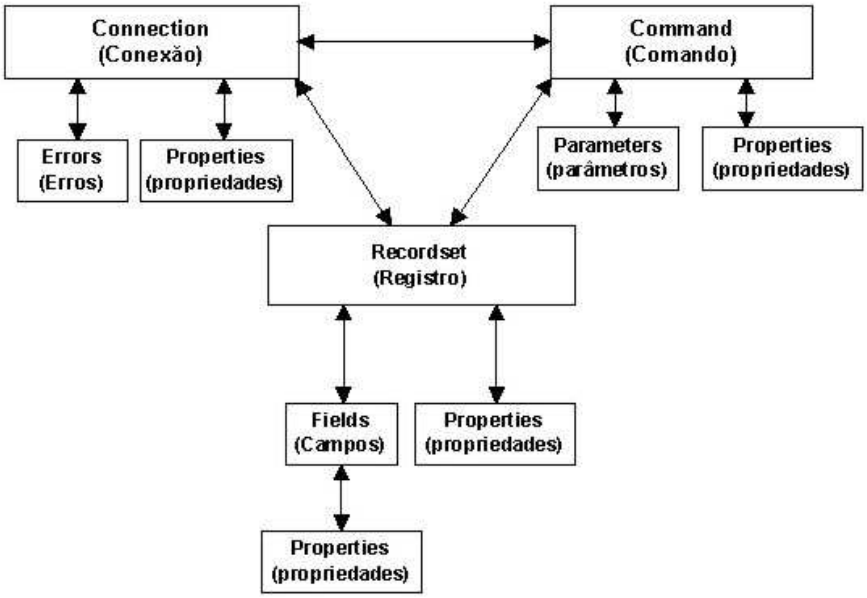
tinham então que vir com um driver ODBC, criava-se um DSN *(Data Source Name)* e assim por diante.

Recentemente a *Microsoft* lançou uma nova tecnologia que substitui o ODBC : O *ActiveX Data Objects* - ADO. No lugar do *driver* surge o *provider*. Então, os DBMS's (Oracle, dBase etc.) terão que vir com *providers* ADO. A idéia básica por trás da criação do ADO é que existam *providers* para tudo que possa se transformar em um *array* de duas dimensões e não só banco de dados relacionais como no ODBC.

Providencialmente, a *Microsoft* criou um *provider* que faz um link (teoricamente) com todos os drivers ODBC. Infelizmente isso não funciona bem para todos os DBMS's. De qualquer maneira este provider ADO-ODBC é o *default* e nem precisa ser citado nas declarações do programa.

O modelo básico do ADO é apresentado na figura 3.

FIGURA 3 - MODELO BÁSICO DO ADO



Assim, para utilizar banco de dados em páginas ASP deve-se instanciar os objetos do ADO ([MAR1999B]) :

a) *Connection* : representa uma conexão ativa com um banco de dados ODBC. Esse é o componente mais importante do ADO, sendo que os demais só poderão ser criados a partir dele;

b) *Command* : representa um comando a ser executado pela fonte de dados ODBC. Com ele pode-se criar comandos SQL compilados no servidor (preparados);
c) *Recordset* : representa um conjunto de registros resultantes do processamento de um comando SQL em um objeto *Connection.* É esse objeto que permite a navegação numa tabela ou consulta do banco de dados;
d) *Fields :* Representa os campos de um RecordSet.

O objeto *ADO Connection* encapsula os objetos fonte de dados do OLE DB (*Data Source*). Ele representa uma única sessão com a fonte de dados. O objeto *Connection* define propriedades da conexão, nomeia o escopo de transações locais, provê uma localização central para tratar erros, e provê um ponto para executar *queries*.

O objeto *ADO Command* é equivalente ao objeto OLE DB *Command*. O *Command* especifica a definição de dados ou declaração de manipulação dos dados a ser executada. No caso de um provedor de dados relacionais, esta é uma instrução SQL. O objeto *Command* permite especificar parâmetros e customizar o comportamento da declaração a ser executada. Um conjunto de objetos *Parameter* expõe esses parâmetros. O objeto *Parameter* representa parâmetros de um comando ([MAR1999B]).

O objeto *ADO Recordset* encapsula o objeto OLE DB *Rowset* (conjunto de linhas, registros). O objeto *Recordset* é a interface atual aos dados, e podem ser dados resultantes de uma *query* ou gerados de outra forma. O objeto *Recordset* provê controle em cima do mecanismo usado, do tipo de cursor usado, do número de linhas para ter acesso de cada vez, e assim por diante. O objeto *Recordset* expõe um conjunto de objetos *Field* que contêm os dados sobre as colunas no *Recordset*, como o nome, tipo, duração, e precisão, como também os próprios dados. *Recordset* é utilizado para navegar por registros de dados.

A interface *Field* representa uma coluna em um *Recordset* que pode ser usado para obter e modificar valores.

A interface *Error* representa um objeto de erro retornado de uma fonte de dados e é opcional, sendo útil quando o provedor de dados subjacente pode retornar erros múltiplos. Se o provedor de dados está impossibilitado de devolver erros múltiplos para uma única chamada de função, o provedor usará o mecanismo COM normal de erros, ou seja, as mensagens de erros normais baseadas na tecnologia COM.

Cada um dos três objetos principais ADO (*Command*, *Connection* e *Recorset*) contêm um conjunto de objetos *Properties*. O objeto *Properties* permite que o ADO exponha dinamicamente as capacidades de um provedor específico. Pelo fato de nem todos os provedores de dados terem a mesma funcionalidade, é importante que o ADO modele isso, para permitir o acesso de modo dinâmico em provedor de dados específicos. Isto também previne que o modelo ADO fique desordenado, cheio de propriedades que só estão disponíveis em certas circunstâncias ([MAR1999B]).

Sem dúvida a classe mais interessante e usada é a *Recordset*. Um objeto *Recordset* vai conter as linhas selecionadas de uma tabela. Na classe *Connection* tem-se o método *Execute* que permite executar códigos SQL.

Para utilizar ADO o primeiro objeto que deve ser instanciado é o *Connection*.

Set Conexao = Server.CreateObject(“ADODB.Connection”)

Após sua definição, é necessário ativar a conexão com a fonte de dados ODBC. Para isso, usa-se o método *Open* do *Connection*.

Sintaxe :

ObjConexao.Open FonteODBC , Usuário, senha

Onde :

a) **FonteODBC :** é o nome da fonte OBDC ou uma string de conexão com o banco;
b) **Usuário** *:* é o nome do usuário da banco de dados (opcional);
c) **Senha** : é a senha desse usuário (opcional).

Exemplo :

Conexao.Open “BlucargoAsp”

O objeto *Connection* possui vários outros métodos, sendo que os principais são apresentados na tabela 11.

TABELA 11 - PRINCIPAIS MÉTODOS DO OBJETO CONNECTION

Close	fecha a conexão e libera recursos no Servidor de banco de dados
Execute (ComandoSQL)	executa um comando SQL *(Structure Query Language)* na conexão aberta. Quando o comando SQL for de seleção, o método retorna um objeto do tipo *RecordSet* com o resultado da consulta. Quando o comando SQL for de execução esse método

	não retorna nada
BeginTrans	inicia uma transação no banco de dados
CommitTrans	finaliza com sucesso uma transação iniciada
RollBackTrans	desfaz todos os comandos de uma transação aberta

Para instanciar um *RecordSet* foi apresentado um exemplo no quadro 13.

QUADRO 13 - EXEMPLO DE INSTANCIAÇÃO DO OBJETO RECORDSET

Set Tabela = Server.CreateObject(“ADODB.RecordSet”)

Em seguida deve-se indicar a fonte de dados ODBC para o *Recordset* :

Tabela.ActiveConnection = “BlucargoAsp”

Por fim, abre-se o *Recordset* utilizando o método *Open* e informando o comando SQL:

Tabela.Open “Select * From Pessoa”

Mas a forma mais fácil de se instanciar um *Recorset* é pela utilização do método *Execute* do *Connection* :

Set Tabela = Conexao.Execute(“Select * from pessoa”)

Os principais métodos e propriedades do objeto *RecordSet* são (tabela 12).

TABELA 12 - MÉTODOS E PROPRIEDADES DO OBJETO RECORDSET

BOF	*True* indica início do *RecordSet*
EOF	*True* indica o final do *RecordSet*
Close	fecha o *RecordSet* aberto
MoveFirst	move o cursor para o primeiro registro de um *RecordSet*
MoveLast	move o cursor para o último registro de um *RecordSet*
MoveNext	move o cursor para o próximo registro de um *RecordSet*
MovePrevious	move o cursor para registro anterior de um *RecordSet*
Update	salva as alterações feitas no RecordSet no Banco de Dados. Também serve para alterar o registro corrente do *RecordSet* exemplo: *Tabela.Update* “email”,”cpd.bluc@zaz.com.br ”
AddNew	adiciona um registro ao *RecordSet* exemplo: Tabela.AddNew Tabela(“Nome”)=”Rodrigo” Tabela(“DataN”)=#04/01/1978# Tabela.Update
Delete	elimina um registro de um *RecordSet* consequentemente, de uma tabela do banco de dados

Para exibir os valores dos campos de todos os registros de um *RecordSet* usa-se a coleção de objetos *Fields* (quadro 14).

QUADRO 14 - EXEMPLO DE REGISTROS DO OBJETO RECORDSET

Do While not Tabela.EOF

```
Response.Write Tabela.Fields(0).Value & “<BR>”
Response.Write Tabela.Fileds(“Nome”) & “<BR>”
Response.Write Tabela.Fields(“email”) & “<BR>”
Response.Write Tabela.Fields(“DataN”) & “<HR>”
Tabela.MoveNext
Loop
```

O quadro 15 mostra um exemplo da utilização dos objetos *Fields*.

QUADRO 15 - EXEMPLO DE REGISTROS DO OBJETO RECORDSET

```
Exemplo :  BANCO1.ASP
<% @LANGUAGE=VBSCRIPT %>
<% OPTION EXPLICIT %>
<%  DIM CONEXAO,TABELA
  SET CONEXAO = SERVER.CREATEOBJECT(“ADODB.CONNECTION”)
  CONEXAO.OPEN “BlucargoAsp”
  SET TABELA = CONEXAO.EXECUTE (“SELECT * FROM PESSOA”)    %>
<HTML><HEAD><TITLE>BlucargoAsp</TITLE></HEAD><BODY><CENTER>
<H1> RELAÇÃO DE PESSOAS </H1>
<%
Do While not Tabela.EOF
     Response.Write Tabela.Fields(0).Value & “<BR>”
    Response.Write Tabela.Fileds(“Nome ) & “<BR>”
    Response.Write Tabela.Fields(“email”) & “<BR>”
    Response.Write Tabela.Fields(“DataN”) & “<HR>”
     Tabela.MoveNext
Loop
%>
</CENTER></BODY></HTML>
```

Para efetuar operações de inclusão, deleção ou modificação num banco de dados utilizam-se comandos SQL do tipo INSERT, DELETE e UPDATE, respectivamente, no método *Execute* do *Connection* (quadros 16,17 e 18).

QUADRO 16 - EXEMPLO DO COMANDO DELETE NO OBJETO CONNECTION

```
BANCO2.ASP – APAGAR
<% @LANGUAGE=VBSCRIPT %>
<% OPTION EXPLICIT %>
<%  DIM CONEXAO,SQL
SET CONEXAO = SERVER.CREATEOBJECT(“ADODB.CONNECTION”)
CONEXAO.OPEN “BlucargoAsp”
SQL=“DELETE FROM PESSOA WHERE CODIGO=” & REQUEST(“CODIGO”)
CONEXAO.EXECUTE (SQL) %>
<HTML><HEAD><TITLE>BlucargoAsp</TITLE></HEAD><BODY><CENTER>
Pessoa de Código igual a <%=REQUEST(“CODIGO”)%> foi excluida do Banco!
</CENTER></BODY></HTML>
```

O quadro 17 mostra um exemplo do operador de modificação UPDATE no método *Execute* do *Connection.*

QUADRO 17 - EXEMPLO DO COMANDO UPDATE NO OBJETO CONNECTION

```
BANCO3.ASP – ALTERAR
<% @LANGUAGE=VBSCRIPT %>
<% OPTION EXPLICIT %>
<%  DIM CONEXAO,SQL
SET CONEXAO = SERVER.CREATEOBJECT("ADODB.CONNECTION")
CONEXAO.OPEN "BlucargoAsp"
SQL = "UPDATE PESSOA SET "
SQL = SQL & "EMAIL = '" & REQUEST("EMAIL") & "' "
SQL = SQL & "WHERE CODIGO=" & REQUEST("CODIGO")
CONEXAO.EXECUTE (SQL) %>
<HTML><HEAD><TITLE>BlucargoAsp</TITLE></HEAD><BODY><CENTER>
Novo email da pessoa <%=REQUEST("CODIGO")%>é<%=REQUEST("email")%>
</CENTER></BODY></HTML>
```

O quadro 18 mostra um exemplo do operador de inserção INSERT no método *Execute* do *Connection.*

QUADRO 18 - EXEMPLO DO COMANDO INSERT NO OBJETO CONNECTION

```
Exemplo BANCO4.ASP   -  INSERIR
<% @LANGUAGE=VBSCRIPT %>
<% OPTION EXPLICIT %>
<%  DIM CONEXAO,SQL
  SET CONEXAO = SERVER.CREATEOBJECT("ADODB.CONNECTION")
 CONEXAO.OPEN "BlucargoAsp"
 SQL = "INSERT INTO PESSOA(NOME,EMAIL,DATAN)VALUES("
 SQL = SQL & "'" & REQUEST("NOME") & "',"
 SQL = SQL & "'" & REQUEST("EMAIL") & "',"
 SQL = SQL & "#" & REQUEST("DATAN") & "#)"
 CONEXAO.EXECUTE (SQL) %>
<HTML><HEAD><TITLE>Ex. de  ASP</TITLE></HEAD><BODY><CENTER>
 <%=REQUEST("NOME")%> foi inserido(a) no Banco!
</CENTER></BODY></HTML>
```

No quadro 19 é apresentado o código da página FORMBANCO.asp que contém formulários para acessar as páginas anteriores.

QUADRO 19 - EXEMPLO DE ACESSO AOS QUADROS ANTERIORES

```
FORMBANCO.asp
<% @ LANGUAGE= VBSCRIPT %>
<HTML><HEAD><TITLE>Exemplo de ASP</TITLE></HEAD>
<BODY><CENTER>
<H1>INSERIR USUÁRIO</H1>
<FORM ACTION="Banco4.asp" METHOD=GET>
  Nome do usuário <BR>
  <INPUT TYPE=TEXT NAME="Nome" MAXLENGTH=50><BR>
  Email do usuário <BR>
 <INPUT TYPE=TEXT NAME="email" MAXLENGTH=50><BR>
Data de Nascimento <BR>
<INPUT TYPE=TEXT NAME="datan" MAXLENGTH=10 VALUE =
"<%=DATE%>"><BR>
<INPUT TYPE=SUBMIT VALUE="INSERIR"><BR>
```

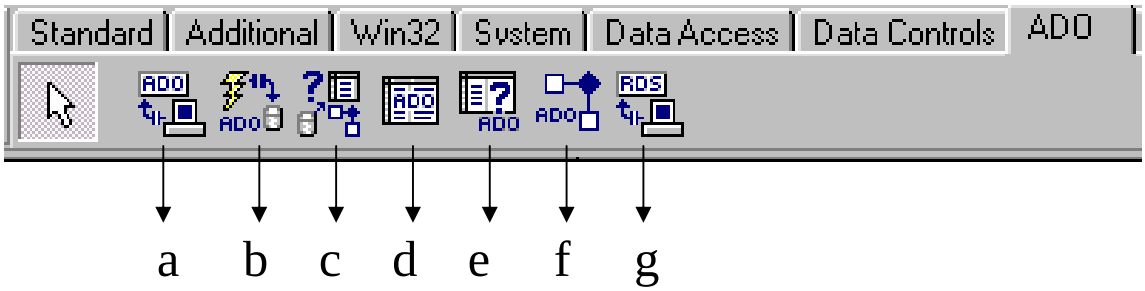
```
</FORM>
<HR>
<H1>APAGAR USUÁRIO</H1>
<FORM ACTION="Banco2.asp" METHOD=GET>
Código do usuário <BR>
<INPUT TYPE=TEXT NAME="Codigo" MAXLENGTH=5><BR>
<INPUT TYPE=SUBMIT VALUE="APAGAR"><BR>
</FORM>
<HR>
<H1>ALTERAR EMAIL</H1>
<FORM ACTION="Banco3.asp" METHOD=GET>
  Código do usuário <BR>
  <INPUT TYPE=TEXT NAME="Codigo" MAXLENGTH=5><BR>
  E-mail do usuário <BR>
  <INPUT TYPE=TEXT NAME="email" MAXLENGTH=50><BR>
  <INPUT TYPE=SUBMIT VALUE="MODIFICAR"><BR>
</FORM>
<HR>
</CENTER>
</BODY>
</HTML>
```

3.3 DELPHI COM ADO

A linguagem de programação Delphi foi utilizada na implementação do protótipo. O Delphi, até sua versão 4, só acessava banco de dados através do BDE *(Borland DataBase Enginee)*. Mesmo quando era necessário utilizar o ODBC, a camada BDE servia como intermediária de comunicação. Com o advento da programação para Internet e a difusão do desenvolvimento em múltiplas camadas, a *Inprise*, empresa que desenvolve o Delphi, viu a necessidade de estender o padrão de acesso, visto que nem todos os provedores ou camadas intermediárias da aplicação poderão ter o BDE instalado. Sendo assim, o Delphi 5 permite o acesso também via ADO *(ActiveX Data Objects)*, de forma muita parecida com o acesso via BDE ([CAN1999]).

O Delphi na sua versão 5 incorpora uma biblioteca de 7 componentes para acesso à camada ADO (figura 6).

FIGURA 4 - PALETA DE COMPONENTES ADO



a) ADOConnection : monta uma conexão persistente para um banco de dados ADO e fornece apoio a transações;
b) ADOCommand : emite um comando SQL diretamente para um banco de dados ADO sem voltar um resultado;
c) ADODataSet : representa os dados de uma ou mais tabelas em um banco de dados ADO e permite que componentes manipulem estes dados por um *DataSource*, e pode estar utilizando ADOTable, ADOQuery, ou ADOStoredProc;
d) ADOTable : representa os dados de uma tabela de banco de dados única via ADO e permite que componentes manipulem este dados por um *DataSource*;
e) ADOQuery : usa declarações do SQL para recuperar dados de uma tabela de banco de dados física via ADO e permite manipular estes dados por um *DataSource*;
f) ADOStoredProc : habilita uma aplicação para acessar um servidor usando ADO;
g) RDSConnection : administra o "marshaling" de dados quando um objeto do Recordset é passado de um processo ou máquina para outra.

A camada ADO é um novo padrão *Microsoft* para acesso a banco de dados ([CAN1999]). A camada ADO são classes de alto nível para acesso ao OLE DB.

Se um usuário de um sistema distribuído que utiliza ADO no acesso aos dados quiser mudar sua fonte de dados, não terá problemas, basta fazer as devidas mudanças na conexão dos dados com os componentes ADO. As camadas podem estar fisicamente separadas, cada

qual numa máquina distinta, ou todas as camadas numa única máquina. A camada de dados também independe da localização.

Esta tecnologia é considerada apropriada para o desenvolvimento desse protótipo devido ao fato da a camada ADO ser específica para aplicações Internet e o COM uma tecnologia que define um modo padronizado para um módulo cliente e um módulo servidor, criando uma interface específica entre os módulos desenvolvidos.

A comunicação entre as camadas é realizada pela tecnologia que implementa objetos distribuídos, chamada DCOM *(Distributed Component Object Model)*, que é uma extensão do COM *(Component Object Model)* ([MIC1999A]). Esta tecnologia será explicitada no próximo capítulo.

4 SISTEMAS DISTRIBUÍDOS

Segundo [MAR1999B], considera-se um sistema distribuído aquele que se mantém em perfeito estado de funcionamento, apesar de seus dados e/ou programas estarem geograficamente distantes (distribuídos).

Em um projeto de sistema distribuído, embora os dados e/ou programas estejam geograficamente distantes, a visão do usuário é de que os mesmos estão ali, junto dele (como se o acesso fosse local). A construção de um projeto distribuído implica na escolha de um meio pelo qual se conseguirá reunir os dados em um determinado momento, para atender por exemplo, a uma consulta feita pelo sistema.

Para efetivação de um projeto de sistema distribuído é necessário a utilização de um software de desenvolvimento que permita, facilite e gerencie a distribuição. A este aspecto soma-se a necessidade de um hardware que evite o máximo possível o estrangulamento local de comunicação (por exemplo modens, *hubs*, *switch*, roteadores, etc), visto que os meios de tráfego externo das mesmas já será lento.

Atualmente os principais métodos de implementação de sistemas distribuídos são o COM/DCOM da *Microsoft*® e o CORBA da OMG, que utiliza como *middleware* o ORB (*Object Request Broker*).

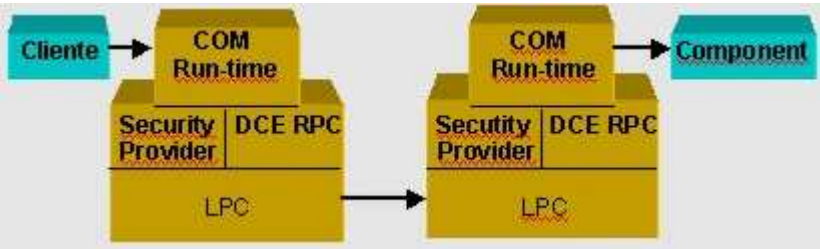
4.1 COM/DCOM

O COM foi criado pela *Microsoft* em 1993 e pode ser considerado como sendo uma tecnologia de empacotamento, um grupo de convenções de bibliotecas de suporte que possibilita a interação entre diferentes partes de software de uma maneira consistente e orientada a objeto. É possível escrever objetos COM em diferentes linguagens como C++, Java, Visual Basic e Delphi entre outras, e eles podem ser implementados em DLL *(Dymamic Link Libraries)* ou nos próprios executáveis, rodando como processos distintos. Um cliente usando COM não precisa ter conhecimento sobre qual a linguagem em que o objeto foi escrito ou se está rodando em uma DLL ou num processo separado ([SES1998]).

No entanto, a primeira versão do COM assumia que os objetos e os clientes estavam rodando na mesma máquina. Desde o inicio, entretanto, os projetistas do COM tiveram a intenção de possibilitar que os clientes criassem e acessassem os objetos em outras máquinas.

A FIGURA 5 mostra a comunicação entre dois objetos utilizando o modelo COM. Um cliente faz a requisição a um servidor de objetos e essa requisição é padronizada pelas camadas do COM, passando por mecanismos de segurança até o DCE *(Distributed Computing Environment)* e RPC *(Remote Procedure Calls)* que se comunica com o servidor através do LPC *(Local Procedure Calls).* O servidor efetua a operação inversa até o componente requisitado.

FIGURA 5 - ARQUITETURA COM



Fonte : [MIC1999A]

4.1.1 APLICAÇÕES COM

Considera-se aplicações do COM, programas correntes que definem uma tarefa ou um processo, seja ele cliente ou servidor. Essas aplicações têm três responsabilidades específicas para assegurar operações próprias com outros componentes :

a) uma aplicação de inicialização deve verificar se a versão da biblioteca COM é atualizada o suficiente para suportar a funcionalidade esperada pela aplicação. Em geral, uma aplicação pode usar uma versão atualizada da biblioteca, mas não uma antiga ou uma outra que tenha atualizado a troca da versão principal;
b) uma aplicação de inicialização deve inicializar a biblioteca COM;
c) uma aplicação de finalização deve finalizar a biblioteca COM.

Para que essas responsabilidades sejam cumpridas, a biblioteca COM deve dar suporte para as aplicações.

4.1.2 CLIENTES COM

Um cliente COM é qualquer peça de código que faça uso de outro objeto através de interfaces de objetos. Desta forma, um cliente COM pode ser também um servidor COM. ([MAR1999A]).

O cliente pode chamar uma função específica para criar um objeto, ou pode solicitar um objeto existente para criar outro, ou ele mesmo pode implementar um objeto para o qual algum outro código transmita ainda outros indicadores de interface de objetos.

Em qualquer caso, qualquer código de cliente usa um CLSID *(Class Identifier)* que, geralmente, irá desempenhar a seguinte seqüência de operações para fazer uso de um objeto :

a) deverá identificar a classe do objeto que será usado;
b) deverá obter a C*lass Factory* (fábrica de classes) para a classe do objeto e solicitar a criação de uma condição de desinstalação da classe de objetos, retornando um indicador de interface para isso;
c) inicializará o objeto criado recentemente, chamando uma função membro de inicialização de uma interface de inicialização, isto é, um dos pequenos conjuntos de interfaces que têm tais funções;
d) fará uso de um objeto, o qual geralmente inclui chamadas *QueryInterface* (manipula a informação e a compatibilidade de tipo de objetos), para obter trabalhos adicionais de indicadores de interfaces em um objeto. Entretanto, o cliente deverá estar preparado para a ausência potencial da interface desejada;
e) deverá soltar o objeto quando ele não for mais necessário.

Após o objeto ser criado e o cliente ter seu primeiro indicador de interface para esse objeto, o cliente não possui meios de saber se o objeto está em processo, se o objeto é local ou remoto, quando examinar o identificador de interface ou qualquer outra interface sobre esse objeto. Isto é, todos os objetos aparecem identicamente para um cliente, tal qual após a sua criação, todos os requerimentos feitos para os serviços de objeto são feitos por chamadas a funções membro da interface ([MAR1999A]).

A biblioteca COM assegura que a chamada feita a um objeto local ou a um objeto remoto seja, de fato, apropriadamente direcionada para outro processo ou outra máquina, respectivamente. Essa operação é transparente para o cliente, que sempre vê qualquer chamada para um objeto como uma chamada à função para as interfaces de objetos, como se esse objeto estivesse em processo. Essa consistência é o benefício chave para os clientes COM, uma vez que lhes garante tratar todos os objetos identicamente, independentemente de seu contexto de execução atual.

4.1.3 SERVIDORES COM

Um servidor COM é qualquer módulo de código, uma DLL ou um EXE, que implemente uma ou mais classes de objetos (cada uma com seu próprio CLSID).

Além disso, os servidores COM também podem ser clientes de outros objetos, usualmente quando o servidor está usando esses outros objetos para ajudar a implementar partes de seus próprios objetos ([MIC2000]).

Se o servidor é uma aplicação, isto é, um programa executável, então ele deve seguir todos os requerimentos para uma aplicação COM. Se o servidor é uma DLL, isto é, um servidor em processamento ou um transmissor de objeto, ele deve pelo menos verificar a versão da biblioteca e pode, se desejar, assegurar que a biblioteca COM seja inicializada. Desse modo, todo os servidores geralmente executam a seguinte seqüência de operações para expor suas implementações de objetos :

a) devem alocar um identificador de classe, um CLSID, para cada classe suportada e prover o sistema com um mapeamento entre o CLSID e o módulo servidor;
b) devem implementar um objeto fábrica de classe com a interface *IClassFactory* (objeto usado para criar uma ou mais instâncias do CLSID) para cada CLSID suportada;
c) devem expor a fábrica de classes tal que a biblioteca COM possa localizá-la depois de carregar (DLL) ou iniciar (EXE) o servidor;
d) devem prover o descarregamento do servidor quando não existirem objetos sendo servidos.

Até onde diz respeito a um servidor, a biblioteca COM existe para dirigir a fábrica de classes do servidor, para criar objetos e para transmitir chamadas de métodos remotos de clientes em outros processos ou em outras máquinas e para coordenar o retorno de valores dos objetos de volta para o cliente ([SES1998]).

4.1.4 INTERFACE REMOTA

No COM, os clientes se comunicam com os objetos somente através do uso de instâncias de interfaces baseadas em *virtual method table* (VMT), também conhecida como *vtable*. O estado do objeto é manipulado pela invocação de funções nessas interfaces. Para

cada método de interface, o objeto provê uma implementação que faz a manipulação apropriada dos objetos internos.

A interface remota provê a infra-estrutura e o mecanismo para permitir uma invocação de método, para que se possa, então, retornar um indicador de interface para um objeto que está em um processo diferente, talvez em uma máquina diferente. A infra-estrutura que permite o isolamento de interface é transparente tanto para o cliente, quanto para o servidor ([MIC1999B]). Nem o cliente, nem o servidor de objeto, são necessariamente conscientes de que a outra parte está de fato em um processo diferente.

4.1.5 A BIBLIOTECA COM

O COM envolve alguns códigos de nível de sistema, isto é, algumas implementações próprias. Todavia, o próprio núcleo do *Component Object Model* (COM) é uma especificação que permite que os objetos e os seus clientes interajam através de padrões binários de interface. Como uma especificação, ele define um número de outros padrões para interoperabilidade :

a) processo fundamental de negociação de interface através de *QueryInterface*;
b) um mecanismo de cálculo de referência através de objetos (e de suas fontes) é gerenciado mesmo quando conectado com clientes múltiplos;
c) regras para alocação de memória e responsabilidade para essas alocações, quando trocadas entre componentes desenvolvidos independentemente;
d) ricas e consistentes facilidades de correspondências de erro.

Em geral, somente um fabricante necessita, ou devia, implementar uma biblioteca COM para qualquer sistema de operação particular. Por exemplo, a *Microsoft* implementou o COM no Windows 3.1, Windows 95, Windows NT e no Apple Macintosh ([MIC1999B]). A implementação dá-se através de uma biblioteca (como a DLL do Windows) que inclui :

a) um pequeno número de funções API *(Application Program Interface)* fundamentais que facilita a criação de aplicações COM, de clientes e servidores. Para os clientes, o COM supre as funções de criação de objetos básicos e para os servidores ele facilita a exposição de seus objetos;
b) serviços de localização de implementação, através dos quais determina, a partir do identificador de classe, que servidor implementa aquela classe e onde aquele servidor se localiza;

c) procedimentos remotos transparentes que verificam quando um objeto está rodando em um servidor local ou remoto. Isto inclui a implementação de um protocolo eletrônico de rede padrão;
d) um mecanismo padrão que permita que uma aplicação controle a alocação de memória dentro do processo.

4.1.6 PROTOCOLO DE REDE COM

O protocolo de rede COM é um protocolo para chamadas de procedimentos remotos orientados a objeto que também é comumente chamado de Objeto RPC *(Remote Procedure Call)* ou ORPC. O protocolo do Objeto RPC compõe-se de um conjunto de extensões ordenadas em camadas sobre um ambiente de computação distribuído (DCE - *Distributed Computing Environment*) ([MIC1999B]).

O protocolo do Objeto RPC especifica :

a) como são feitas as chamadas para um objeto;
b) como são apresentadas, comunicadas e mantidas as referências de objeto.

4.1.7 GERENCIADOR DE CONTROLE DE SERVIÇO

O Gerenciador de Controle de Serviço (*Service Control Manager* - SCM) é um componente da biblioteca COM responsável pela localização de classes da implementação e por rodá-las. O SCM garante que quando um cliente faz uma requisição, o servidor apropriado é conectado e preparado para receber a solicitação. O SCM mantêm uma base de dados de informação de classe baseadas no registro do sistema que o cliente armazena localmente através da biblioteca COM.

Para assegurar a conectividade, serviços SCM localizam-se no exportador de objeto COM em cada máquina. É notável que diferente do serviço exportador de objeto, o qual é solicitado por uma máquina para expor objetos COM remotamente, o serviço SCM exposto é de fato opcional, e algumas máquinas podem não oferecê-los.

4.1.8 CONHECENDO O DCOM

Como o COM já possibilitava a comunicação de aplicativos numa mesma máquina, o próximo passo era conseguir a comunicação de aplicativos em uma rede. Esse papel foi atribuído ao *Distribuited Component Object Model* (DCOM). Embora o COM tenha sido

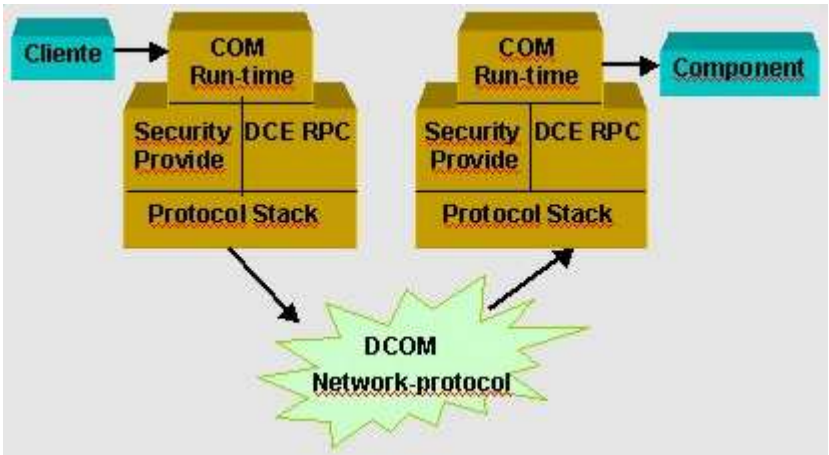
lançado em 1993 o DCOM foi lançado somente em 1995 com o lançamento do Windows NT 4.0. O DCOM implementa um mecanismo de segurança distribuído oferecendo autenticação e encriptação de dados ([MIC1999B]).

Devido o DCOM ser nativo do ambiente Windows, ele possui um total domínio desta plataforma. Quando se utiliza o DCOM se fala em *ActiveX*, então daí vem o domínio das plataformas Windows, mas o DCOM já não é mais uma exclusividade do ambiente Windows. A *Microsoft* está trabalhando em conjunto com a *Bristol Technology* e a *Mainsoft* para dar suporte ao *ActiveX* na plataforma *Unix* e também uniu-se a *Metrowerks* e a *Macromedia* para fazer o mesmo na plataforma Macintosh, sendo que já existe disponível no mercado uma versão do *ActiveX SDK* para Mac.

A comunicação de objetos utilizando DCOM é tão segura quanto a comunicação de objetos utilizando o COM. Ela é totalmente transparente ao programador ou usuário ([MIC1999B]).

A seguir, a figura 6 mostra a comunicação entre dois objetos em rede utilizando o modelo DCOM. Um cliente faz a requisição a um servidor de objetos em outro ponto da rede. Essa requisição é padronizada pelas camadas do COM, passando por mecanismos de segurança até o DCE RPC. Este comunica-se com o servidor através do protocolo *Stark*. O servidor efetua a operação inversa até o componente requisitado.

FIGURA 6 - ARQUITETURA DCOM



Fonte : [MIC1999A]

O DCOM oferece a uma organização muitos benefícios que vão além da criação de aplicações em rede.

4.1.9 GERÊNCIA DE CONEXÕES

O DCOM utiliza um contador de referência para determinar quantos clientes estão conectados a um determinado servidor. Assim, quando um novo cliente se conecta ao servidor, o contador se incrementa e, caso contrário, quando um cliente se desconecta do servidor de aplicações, o contador sofre um decréscimo. Quando o contador de referência chega a zero, a aplicação se encerra. Quando um servidor de aplicações não mais se encontra em funcionamento, o DCOM deixa a aplicação e estabelece o contador de referência da maneira mais adequada ([MIC1999A]).

Se um cliente se desconecta sem que o contador de referência seja decrementado, deve existir um modo para que o servidor decremente o contador de referência. Isto é feito enviando um *ping* (sinal enviado ao destinatário verificando se o mesmo está ativo) do servidor para o cliente. Se o servidor não conseguir enviar um *ping* três vezes seguidas, a conexão será considerada encerrada e o contador de referência será ajustado.

4.1.10 INTERFACES MÚLTIPLAS

O DCOM é implementado fundamentalmente com a utilização de interfaces. As interfaces são os métodos expostos e as propriedades de um objeto em particular. Podem haver muitas interfaces para um mesmo objeto. Isto oferece a oportunidade de personalizar um objeto de acordo com as necessidades do programador, sem ter a necessidade de reconstruí-lo. Isso é muito importante quando um objeto requer uma funcionalidade adicional que somente necessita de um subconjunto dos clientes. Nessa situação, o programador pode utilizar uma mesma implementação com diferentes interfaces. Uma interface pode proporcionar uma funcionalidade básica necessária e outra pode proporcionar uma funcionalidade melhorada. Isto poderá levar, em ambos os casos, a uma mesma implementação real ([MIC1999A]).

4.1.11 INTEGRAÇÃO COM OUTROS PROTOCOLOS DE REDE

Como o DCOM está baseado nos protocolos *Transmission Control Protocol/Internet Protocolo* (TCP/IP) e *Remote Procedure Call* (RPC), os servidores de aplicação podem ser

invocados desde qualquer ponto da Internet utilizando o TCP/IP básico ou outros protocolos baseados em TCP (como *Point to Point Tunneling Protocol* - PPTP) para usuários que conectam o HTTP da Internet. Suportando o nível mais baixo de interoperabilidade de rede, a comunicação em rede do DCOM é transparente para as aplicações. Isto oferece muitas opções de rede distintas para o desenvolvedor, o qual poderá escolher a melhor conectividade para a aplicação ([MIC1999A]).

4.1.12 INDEPENDÊNCIA NA LOCALIZAÇÃO E EQUILÍBRIO DE PROCESSOS

A independência na localização é o conceito onde uma aplicação cliente não precisa se preocupar em averiguar onde está localizado o servidor de aplicações para poder se conectar ao mesmo. Como o DCOM está dirigido para as aplicações conectadas na rede, o único critério que se deve verificar é se a aplicação realmente existe na rede. O modo como uma aplicação cliente localiza um servidor de aplicações é encontrando as entradas de registro adequadas no arquivo de registro da aplicação cliente. Simplesmente para trocar informação de um servidor A para um servidor B o cliente terá que processar a aplicação no servidor B. Quando uma aplicação tem vários usuários, estes podem se dividir em grupos de usuários. Por exemplo, o grupo 1 poderia utilizar uma aplicação em um servidor A, enquanto que o grupo 2 poderia utilizar a mesma aplicação em um servidor B.

Este tipo de equilíbrio de processo se chama estático, porque os usuários sempre conectam com o mesmo servidor, seja qual for o número de usuários que estejam conectados. Isto ocorre em situações simples, mas na maioria das vezes os processos precisam se equilibrar dinamicamente. Tomando o exemplo acima não tão completo, mas realista, em um certo momento o grupo1 poderá requerer muito mais do servidor de aplicações. Nesse mesmo momento as requisições do grupo2 serão baixadas. Se o equilíbrio de processo for estático, o servidor A estará bastante subutilizado enquanto que o servidor B estará sobrecarregado. Se de alguma forma alguns usuários do grupo1 passarem a utilizar o servidor B ao invés do servidor A, se fosse basear-se no tráfego de rede existente para cada servidor, haverá um equilíbrio na utilização global dos servidores, melhorando a performance da rede. O DCOM por si mesmo não resolve o equilíbrio dinâmico de carga ([RAJ1999]).

4.1.13 SEGURANÇA FRENTE AS FALHAS E TOLERÂNCIA AOS ERROS

O DCOM proporciona suporte básico frente aos erros com as utilidades do *pinging*. Essencialmente, isso significa que se um servidor de aplicações falhar por qualquer razão, o DCOM tentará reinicializar a aplicação. Ainda que esta seja uma função importante, não resolve os momentos em que o servidor de aplicações não pode se reinicializar por diversos motivos, incluindo falhas no fornecimento de energia elétrica, falhas no hardware, erros na aplicação e etc ([MIC1999B]).

Segundo a *Microsoft*, a solução para um sistema de aplicações realmente tolerante à falhas se descreve no texto *"DCOM Technical Overview"*, da seguinte maneira :

"DCOM facilita a implementação de tolerância aos erros. Uma técnica é o componente de referência introduzido na seção anterior. Quando os clientes detectam a falha de um componente, voltam a se conectar com o mesmo componente de referência que fez a primeira conexão. O componente de referência tem informação sobre os servidores que já não estão disponíveis e proporciona automaticamente ao cliente uma nova amostra do componente funcionando em outra máquina".

4.1.14 INTEGRAÇÃO DE BASE DE DADOS

Para compreender as questões sobre integração de base de dados, pode-se estudar os fundamentos do DCOM. O DCOM se baseia em chamadas a procedimentos remotos (RPC). Entretanto, há uma limitação na RPC, no que se refere ao conjunto limitado de tipos de dados suportados. Esses tipos de dados são os tipos simples, por exemplo : binários, inteiros, vetores e *array*. O que não se suporta é a transferência de tipo de dados complexos, como os objetos. Ainda que os dados de uma base de dados entrem na categoria dos tipos de dados simples, os conjuntos de dados não podem entrar. Ainda que essa técnica transporte os dados, a idoneidade para os dados se perde na conversão. É necessário que exista um mecanismo que distribua um conjunto de dados de bases de dados *(DataSets)*, que permita ao usuário examinar e modificar esse conjunto de dados e permita ao usuário enviar essas modificações novamente para o objeto finalmente à base de dados ([RAJ1999]).

Uma função básica de um servidor de base de dados é a integridade (a validação dos dados). Se o programador não realiza codificação adicional no cliente para evitar óbvias

infrações na integridade dos dados, o usuário final só será informado dessa infração das regras uma vez que tenha enviado os dados de volta ao servidor de aplicações ou ao Sistema Gerenciador de Banco de Dados (SGBD) ([RAJ1999]). Isso dará como resultado uma interface de usuário nada fácil de se manejar e tráfego desnecessário na rede. A alternativa é reprogramar as regras no módulo cliente da aplicação. Entretanto, isso tem outros efeitos secundários. Se a regra for modificada, o módulo cliente deve ser atualizado. Ainda que isso pareça ser uma operação simples, onde se tem muitas aplicações clientes distintas que necessitam ser reconstruídas e muitas máquinas clientes para reinstalar o software, isto é na verdade um processo complexo e que requer bastante tempo.

5 DESENVOLVIMENTO DO PROTÓTIPO

Nesse capítulo, será apresentado o protótipo de sistema de importação, quanto às suas características de modelagem, desenvolvimento e funcionamento.

Para a modelagem do protótipo de sistema de importação foi utilizada a linguagem de UML *(Unified Modeling Language)*.

5.1 UML (UNIFIED MODELING LANGUAGE)

A UML é a sucessora de um conjunto de métodos de análise e projeto orientados a objeto (OOA&D). A UML está, atualmente, em processo de padronização pela OMG *(Object Management Group)* ([FUR1998]).

A UML é uma linguagem de modelagem, não um método. Um método pressupõe um modelo de linguagem e um processo. O modelo de linguagem é a notação que o método usa para descrever o projeto e o processo são os passos que devem ser seguidos para se construir o projeto.

O modelo de linguagem é uma parte muito importante do método. Corresponde ao ponto principal da comunicação. Se uma pessoa quer conversar sobre o projeto, com outra pessoa, é através do modelo de linguagem que elas se entendem. Nessa hora, o processo não é utilizado. A UML define uma notação e um meta-modelo. A notação são todos os elementos de representação gráfica vistos no modelo (retângulo, setas, o texto e etc), é a sintaxe do modelo de linguagem. A notação do diagrama de classe define a representação de itens e conceitos tais como : classe, associação e multiplicidade. Um meta-modelo é um diagrama de classe que define de maneira mais rigorosa a notação.

5.2 ESPECIFICAÇÃO DO PROTÓTIPO

Para a especificação, foram utilizados os diagramas de casos de uso, de classes e de seqüência, elaborados na ferramenta de modelagem *Rational Rose* da *Rational®*.

5.2.1 CASOS DE USO

O diagrama de casos de uso tem como objetivo principal descrever a visão externa do sistema e suas interações com o mundo exterior, de maneira consensual entre usuários e desenvolvedores de sistemas.

Foram identificados 4 atores nesse diagrama de casos de uso (figura 7) :

a) usuário : corresponde ao funcionário da Blucargo Transportes que irá efetuar as operações de cadastros, criação do Booking e do conhecimento, e ainda elaborar as cotações requisitadas pelos clientes;
b) cliente : também chamado de consignatário ou importador. Na verdade, são empresas internacionais que importam produtos de outros países;
c) financeiro : também funcionário da Blucargo Transportes, responsável pelas operações financeiras do sistema;
d) agente : intermediários entre a Blucargo Transportes e os clientes exportadores de carga que representam a Blucargo no Brasil e no Mundo. Os agentes recebem uma comissão por cada transação por ele estabelecida, e essa comissão é chamada de contrato de câmbio.

São 11 (onze) os casos de uso primários e secundários identificados neste protótipo (figura 7) :

a) cadastrar Companhias : neste caso o usuário fará as entradas dos dados referentes às companhias aéreas no sistema e a gravação dos mesmos na base de dados. Somente o ator usuário pode cadastrar as companhias;
b) cadastrar clientes : neste caso o usuário fará as entradas dos dados referentes aos clientes importadores e exportadores, e a gravação dos mesmos na base de dados. Somente o ator usuário pode cadastrar os clientes;
c) cadastrar aeroportos : neste caso o usuário fará as entradas dos dados referentes aos aeroportos no sistema e a gravação dos mesmos na base de dados. Somente o ator usuário pode cadastrar os aeroportos;
d) cadastrar agentes : neste caso o usuário fará as entradas dos dados referentes aos agentes no sistema e a gravação dos mesmos na base de dados. Agentes são os intermediários entre a Blucargo e os clientes exportadores de mercadorias que representam a Blucargo no Brasil e no Mundo. Os agentes recebem uma comissão

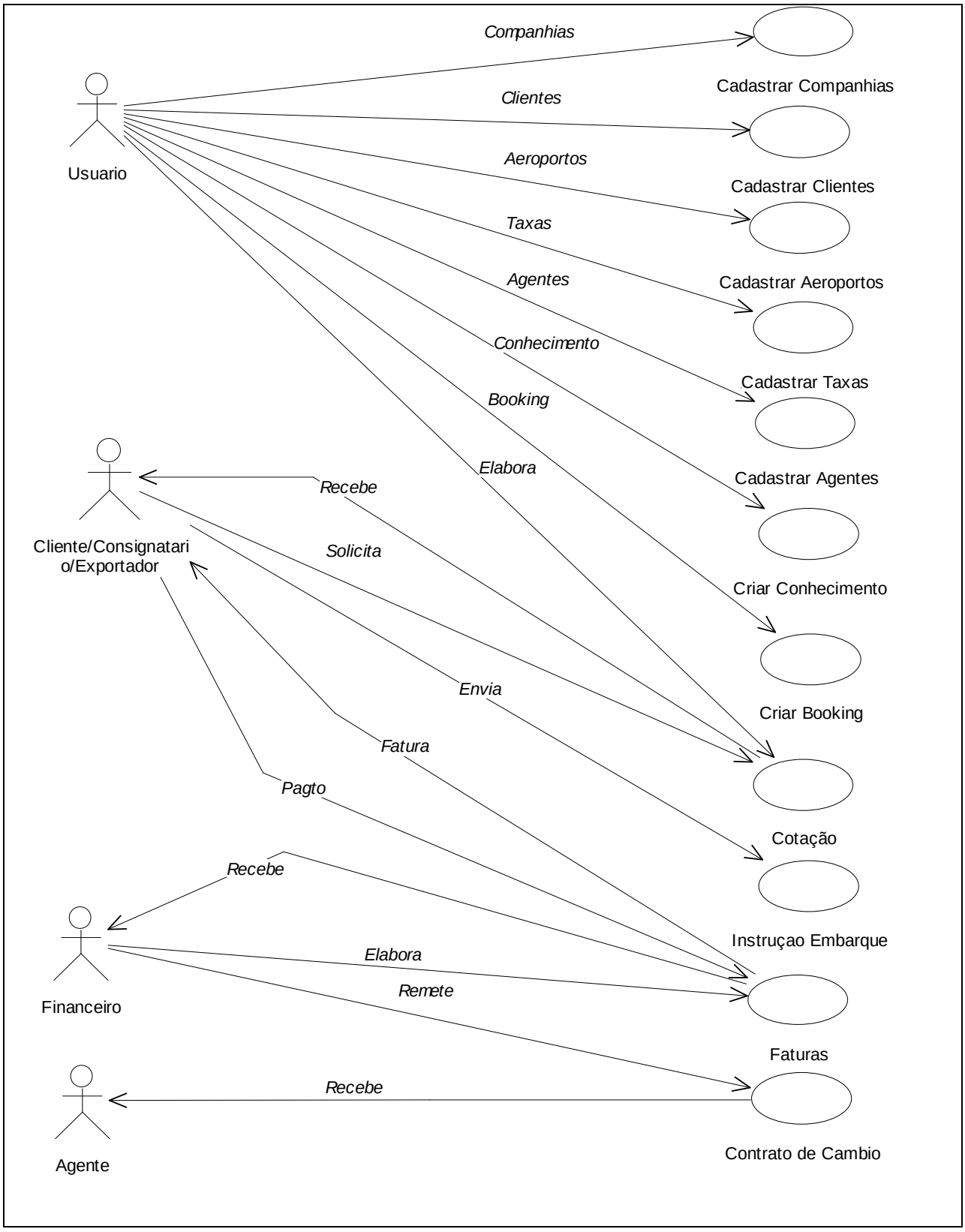
por cada transação por ele estabelecida e essa comissão é chamada de contrato de câmbio. Somente o ator usuário pode cadastrar os agentes;

e) cadastrar taxas : neste caso o usuário fará as entradas dos dados referentes às taxas usadas no processo de importação e a gravação dos mesmos na base de dados. As taxas referem-se à despesas decorridas no trânsito da carga, como por exemplo *delivery fee* (taxa paga à companhia aérea para liberação do conhecimento de embarque original utilizado pelo importador para liberar a carga junto a alfândega brasileira);
f) criar conhecimento : neste caso o usuário fará as entradas dos dados referentes ao conhecimento de carga no sistema e a gravação dos mesmos na base de dados. Conhecimento é o equivalente ao termo de contrato entre o embarcador e o transportador, para o transporte de bens sobre uma rota da transportadora. Os dados do *booking* são necessários para a criação do conhecimento. Somente o ator usuário pode cadastrar os conhecimentos;
g) criar *booking* : neste caso o usuário fará as entradas dos dados referentes ao *booking* no sistema e a gravação dos mesmos na base de dados. A partir do momento que o cliente aceita a cotação e autoriza o embarque, é enviado à Blucargo a fatura comercial do processo de importação, cujas informações serão inseridas no *booking*, juntamente com detalhes da instrução de embarque passada para o agente embarcador da carga no exterior. Somente o ator usuário pode cadastrar o *booking*;
h) cotação : neste caso o usuário fará as entradas dos dados referentes a cotação no sistema e a gravação dos mesmos na base de dados. Cotação é o orçamento que o cliente faz antes de contratar os serviços da Blucargo. O cliente solicita a cotação e a Blucargo envia os valores e taxas sobre determinada carga. Somente o ator usuário pode cadastrar a cotação;
i) instrução de embarque : neste caso o cliente irá enviar uma autorização à Blucargo Transportes, para o embarque da carga importada. A instrução de embarque é o próximo passo depois da cotação, caso seja aceita. É a remessa dos dados da carga a ser transportada;
j) faturas : neste caso o financeiro fará as entradas dos dados referentes as faturas no sistema e a gravação dos mesmos na base de dados. A fatura será enviada ao cliente em forma de cobrança bancária, e posteriormente o mesmo efetuará o pagamento

beneficiando à Blucargo Transportes. As faturas são as cobranças feitas aos clientes depois que todo o processo de importação da carga foi efetuado. Somente o ator financeiro pode fazer as faturas;

k) contrato de câmbio : neste caso o financeiro fará as entradas dos dados referentes ao contrato de câmbio no sistema e a gravação dos mesmos na base de dados. Contrato de câmbio é o pagamento que a Blucargo faz ao agente envolvido com o processo de importação. Somente o ator financeiro pode fazer o contrato de câmbio.

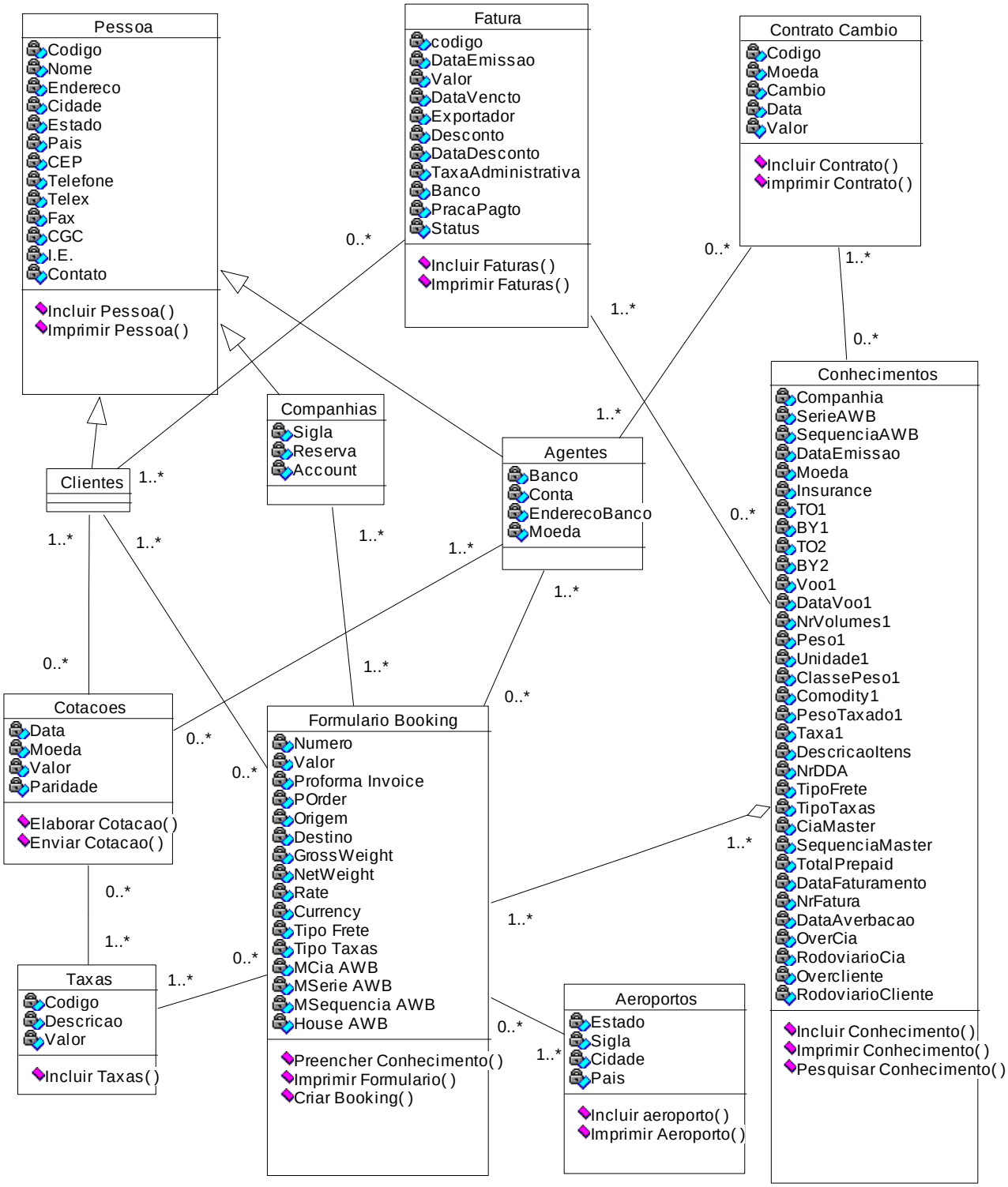
FIGURA 7 - CASOS DE USO



5.2.2 DIAGRAMA DE CLASSES

A figura 8 apresenta o diagrama de classes :

FIGURA 8 - DIAGRAMA DE CLASSES



Existem 11 (onze) classes identificadas no sistema :

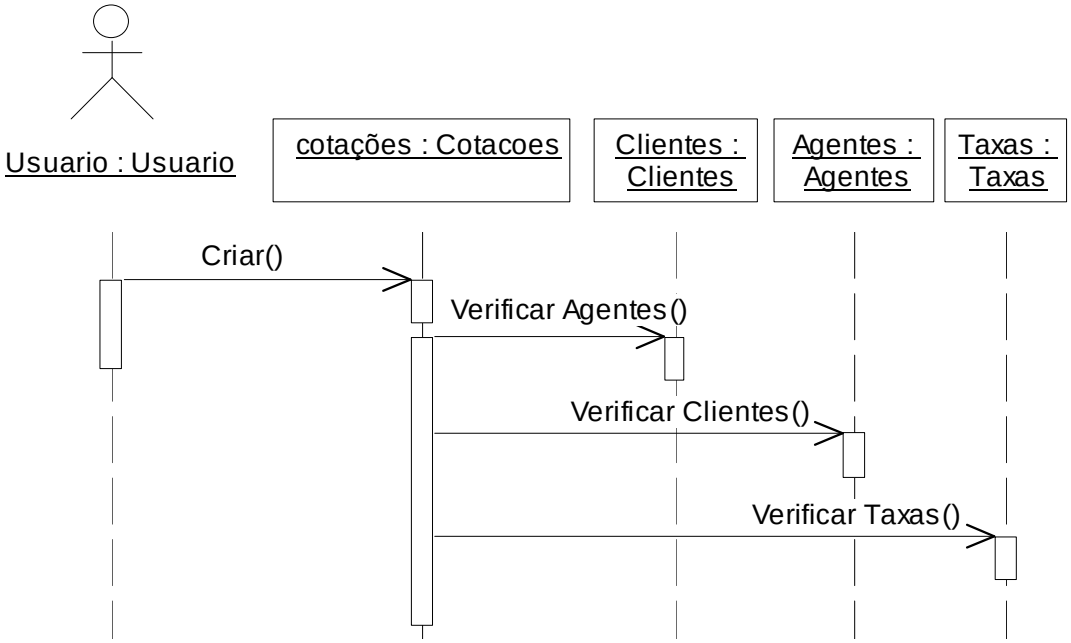
a) pessoa : esta classe é uma classe genérica das classes companhias, agentes e clientes, pelo fato de possuírem atributos em comum;
b) companhias : corresponde às companhias aéreas cadastradas no sistema. A classe *booking* possui informações das companhias envolvidas;
c) agentes : corresponde aos agentes cadastrados no sistema. A classe *booking* possui informações dos agentes envolvidos;
d) clientes : corresponde aos clientes cadastrados no sistema, que podem ser os importadores (consignatários), ou exportadores *(shipper).* Os clientes solicitam as cotações, e cada conhecimento e *booking* possui 1 (um) cliente;
e) conhecimentos : é a classe que contém todas as informações do processo de importação elaborado pelo sistema;
f) *booking* : é a classe que contém todas as informações necessárias para o embarque da carga importada. A classe conhecimentos absorve dados do *booking*;
g) aeroportos : corresponde aos aeroportos cadastrados no sistema. A classe *booking* possui informações dos aeroportos;
h) taxas : corresponde às taxas cadastradas no sistema. O *booking* e as cotações precisam das taxas para calcular valores;
i) cotações : esta classe é uma solicitação do cliente contendo informações sobre valores e taxas da carga a ser importada;
j) contrato de câmbio : é a remessa de capital para os agentes que participaram da importação da carga. Informações do contrato de câmbio são necessárias na classe faturas;
k) faturas : esta classe contém os valores que o cliente deverá remeter junto à Blucargo Transportes.

5.2.3 DIAGRAMAS DE SEQUÊNCIA

Os diagramas de seqüências representam a seqüência em que as ações ocorrem dentro do sistema. Eles demonstram como é feita a troca de mensagens entre as classes. Para cada caso de uso, há um diagrama de seqüência.

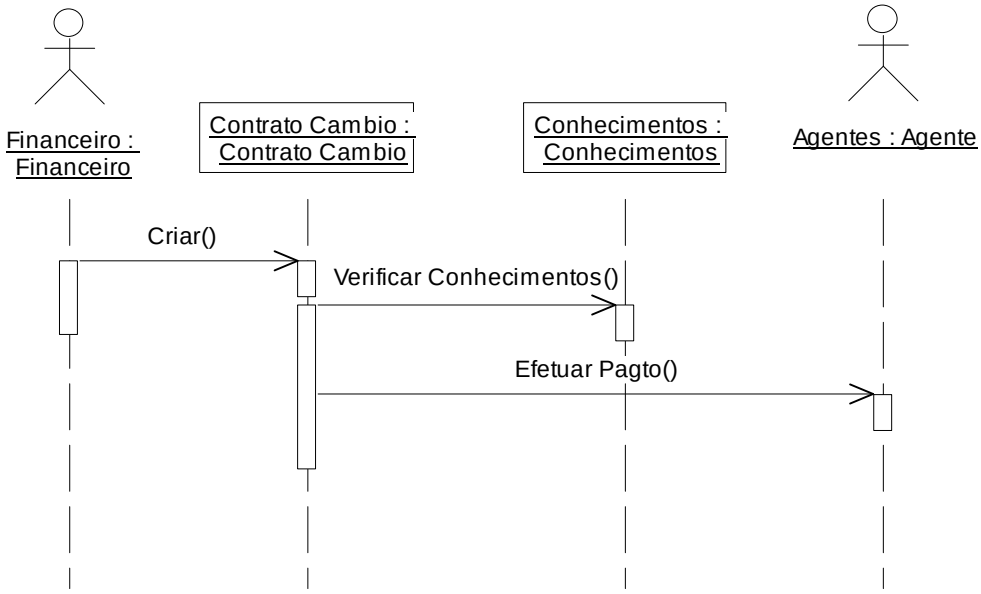
É apresentado na figura 9 o diagrama de seqüência para o caso de uso “cotação”.

FIGURA 9 - DIAGRAMA DE SEQÜÊNCIA COTAÇÃO



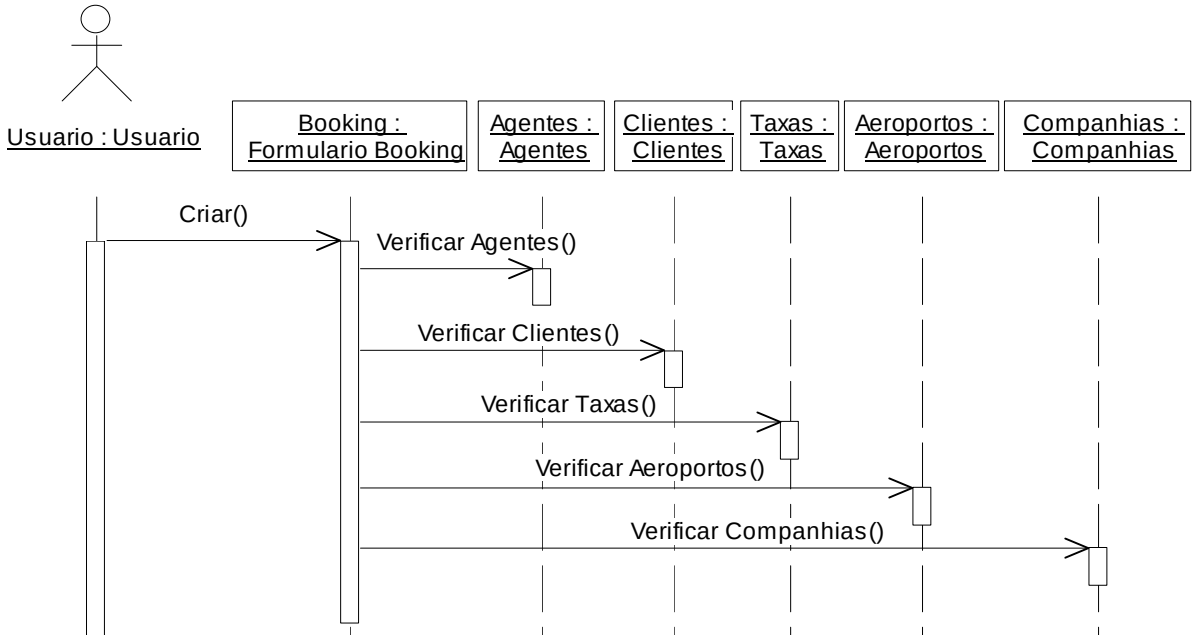
É apresentado na figura 10 o diagrama de seqüência para o caso de uso “contrato de câmbio ”.

FIGURA 10 - DIAGRAMA DE SEQÜÊNCIA CONTRATO DE CÂMBIO



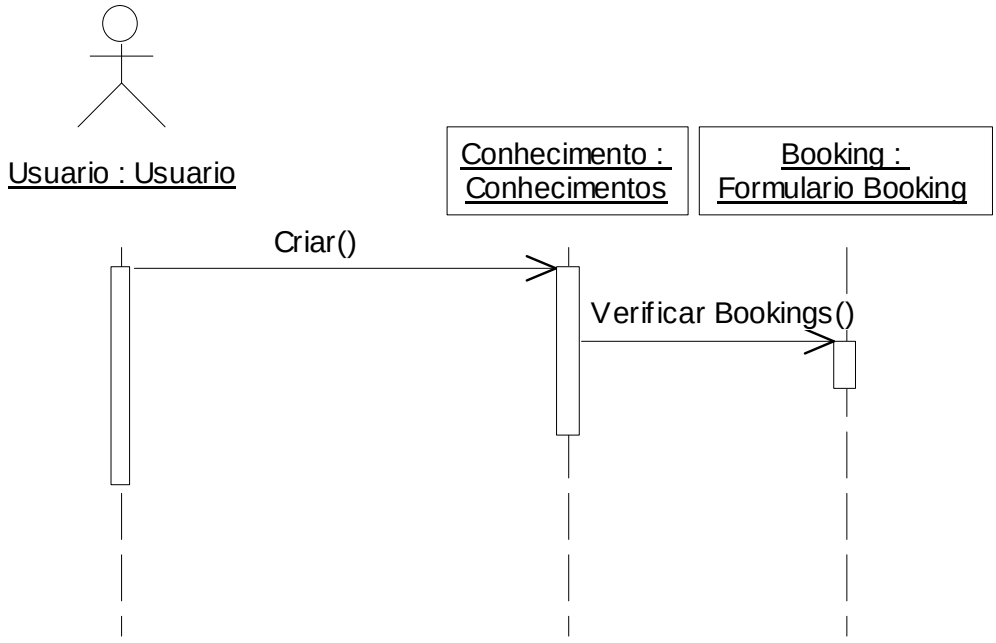
É apresentado na figura 11 o diagrama de seqüência para o caso de uso “*booking* ”.

FIGURA 11 - DIAGRAMA DE SEQÜÊNCIA BOOKING



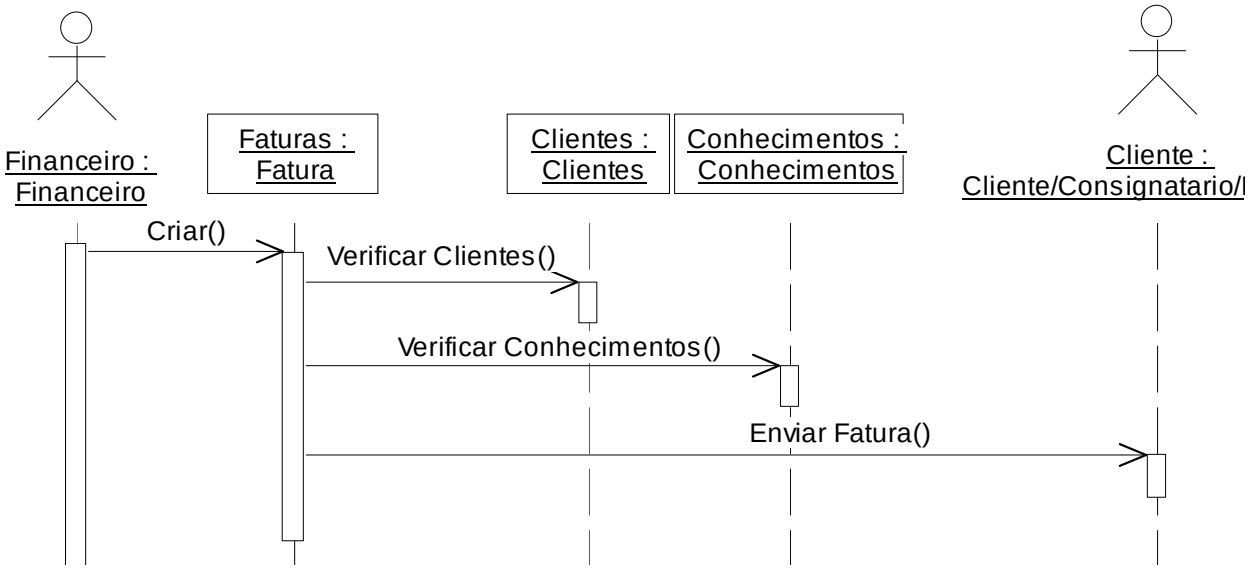
É apresentado na figura 12 o diagrama de seqüência para o caso de uso “conhecimentos”.

FIGURA 12 - DIAGRAMA DE SEQÜÊNCIA CONHECIMENTOS



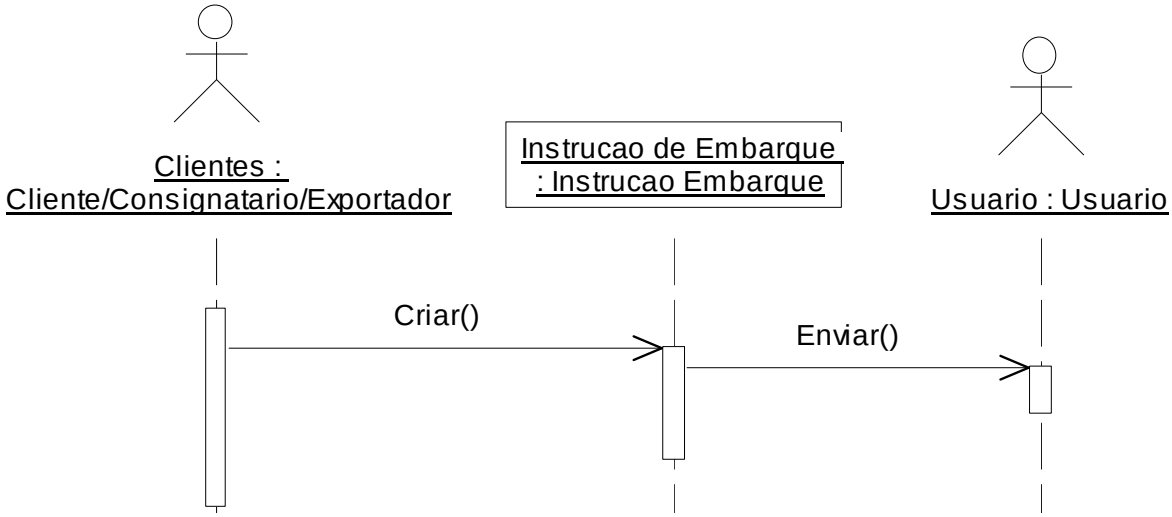
É apresentado na figura 13 o diagrama de seqüência para o caso de uso “faturas ”.

FIGURA 13 - DIAGRAMA DE SEQÜÊNCIA FATURAS



É apresentado na figura 14 o diagrama de seqüência para o caso de uso “faturas ”.

FIGURA 14 - DIAGRAMA DE SEQÜÊNCIA INSTRUÇÃO DE EMBARQUE



5.3 IMPLEMENTAÇÃO

Como a principal característica do protótipo de sistema de importação é a utilização de objetos distribuídos, pode-se dizer que numa visão distribuída, a implementação do protótipo é dividida em três partes ou camadas :

a) camada de aplicação : é o módulo cliente do DCOM, onde funcionará todo o protótipo na visão do usuário do sistema;

b) camada de regras de negócio : é o módulo servidor do DCOM, e essa camada é abstrata para os usuários do protótipo;
c) camada de dados : são os dados do protótipo, formados por um sistema gerenciador de banco de dados (SGBD) e um arquivo de dados. O banco de dados utilizado no protótipo de sistema de importação é o *Microsoft Access*.

A comunicação entre as camadas se dá através da tecnologia DCOM que implementa objetos distribuídos. Existem várias maneiras para se implementar objetos distribuídos. Foram utilizados componentes do Delphi 5.0 que disponibiliza os recursos necessários para a implementação de um programa cliente e um programa servidor, comunicando-se remotamente através de DCOM.

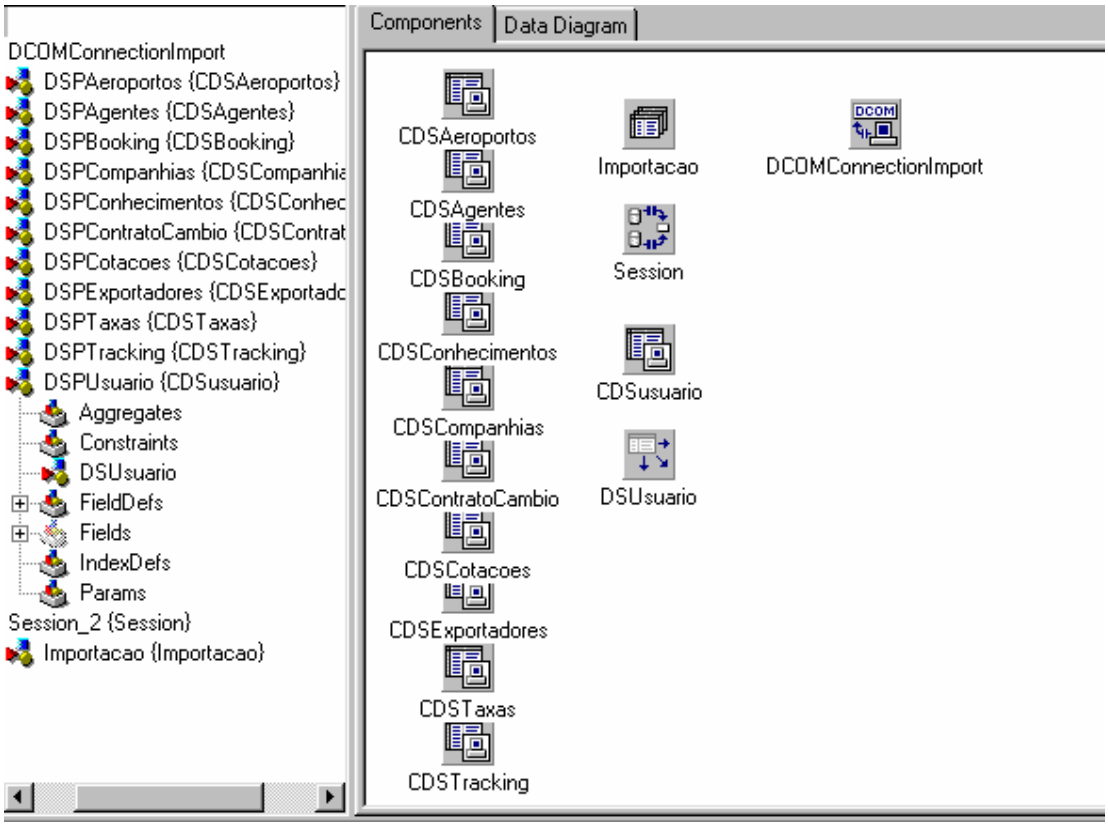
As camadas podem estar fisicamente separadas, cada qual em uma máquina diferente, ou todas em uma única máquina. O programa cliente implementa a camada da aplicação e o programa servidor, a camada de regras de negócios. A camada de dados também independe da localização. Foram aplicados testes de funcionamento do protótipo, utilizando o conceito de equilíbrio de processo estático, onde as aplicações clientes conectam ao mesmo servidor, numa Intranet, mas ambas camadas poderiam estar em qualquer lugar da rede, seja ela uma Intranet ou Extranet (duas ou mais Intranets conectadas utilizando recursos de comunicação da Internet).

5.3.1 CAMADA DE APLICAÇÃO

No camada de aplicação utilizaram-se as técnicas de programação convencionais, porém todo o acesso aos dados é feito utilizando a tecnologia ADO. A independência de fonte de dados que o ADO permite, possibilita que o cliente execute independente do banco de dados.

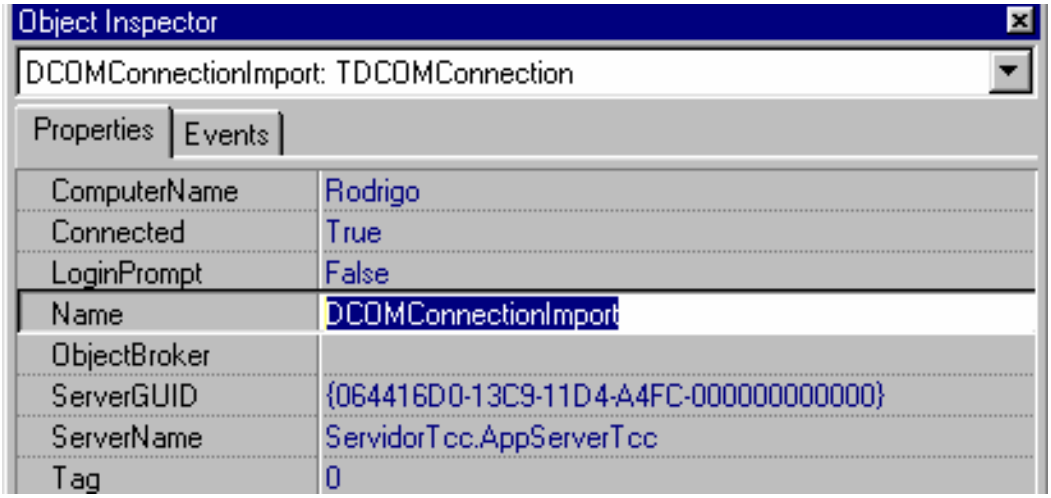
Foi criado no aplicativo cliente do protótipo um módulo de dados *(data module)* simples. A figura 15 demonstra este módulo de dados.

FIGURA 15 - MÓDULO DE DADOS CLIENTE



Para estabelecer a conexão com o servidor DCOM, utilizou-se o componente *TDCOMConnectionImport*, que possui algumas propriedades que devem ser configuradas como visto na figura 16.

FIGURA 16 - PROPRIEDADES DO DCOM CONNECTION



A propriedade *ComputerName* deve conter o nome do servidor. Se estiver em branco, o servidor será a máquina local. A propriedade *Connected*, identifica se a conexão deve ser feita em tempo de projeto. A propriedade *LoginPrompt*, identifica se deve ser pedido um login de acesso antes de conectar ao banco de dados e a propriedade *ServerName*, identifica o servidor DCOM que deverá ser conectado. Ao informar o *ServerName*, automaticamente será preenchida a propriedade *ServerGUID*.

O acesso aos dados no cliente é feito através de componentes *TClientDataSet*. Estes componentes, através da interface *IappServerImportaçãoTCC*, o qual é a interface do servidor DCOM, e comunicam-se com os *DataSetProviders* do módulo de dados remoto, identificado pelo *TDCOMConnection*. Existem duas propriedades principais que devem ser informadas no *TClientDataSet* :

a) *remoteServer* : identifica qual é o servidor remoto. Nesta propriedade é informado o nome da conexão, neste caso, *DCOMConnectionImport*;
b) *providerName* : nome do provedor de dados relacionado, que está no módulo de dados remoto do servidor DCOM.

Para implementação do objeto cliente no protótipo de sistema de importação foram utilizados componentes MIDAS *(Middle-tier Application Services)* do Delphi 5.0. O MIDAS oferece suporte para a maioria dos principais padrões de comunicação distribuída, incluindo DCOM, MTS, Soquetes TCP/IP, HTTP e CORBA. O componente *TDCOMConnection* é um componente MIDAS.

Para a comunicação entre os dois lados do aplicativo (cliente e servidor) é utilizada a interface *IAppServerImportaçãoTCC*. Esta interface tem os seguintes métodos padrões :

a) *as_applyUpdates*: aplica todas as alterações que foram feitas no conjunto de dados que está no cliente, gravando-as na tabela física que se encontra na camada de dados;
b) *as_getRecords*: retorna os registros de um conjunto de dados. Um componente *TClientDataSet*, por exemplo, utiliza este método quando é ativado;
c) *as_execute*: permite que o conjunto de dados cliente interaja com *queries* ou *stored procedures* remotamente. Executa um comando;
d) *as_dataRequest*: é uma requisição de dados do conjunto de dados;
e) *as_getProviderNames*: retorna uma lista de todos os servidores compatíveis que

estão em um Módulo de Dados Remoto;

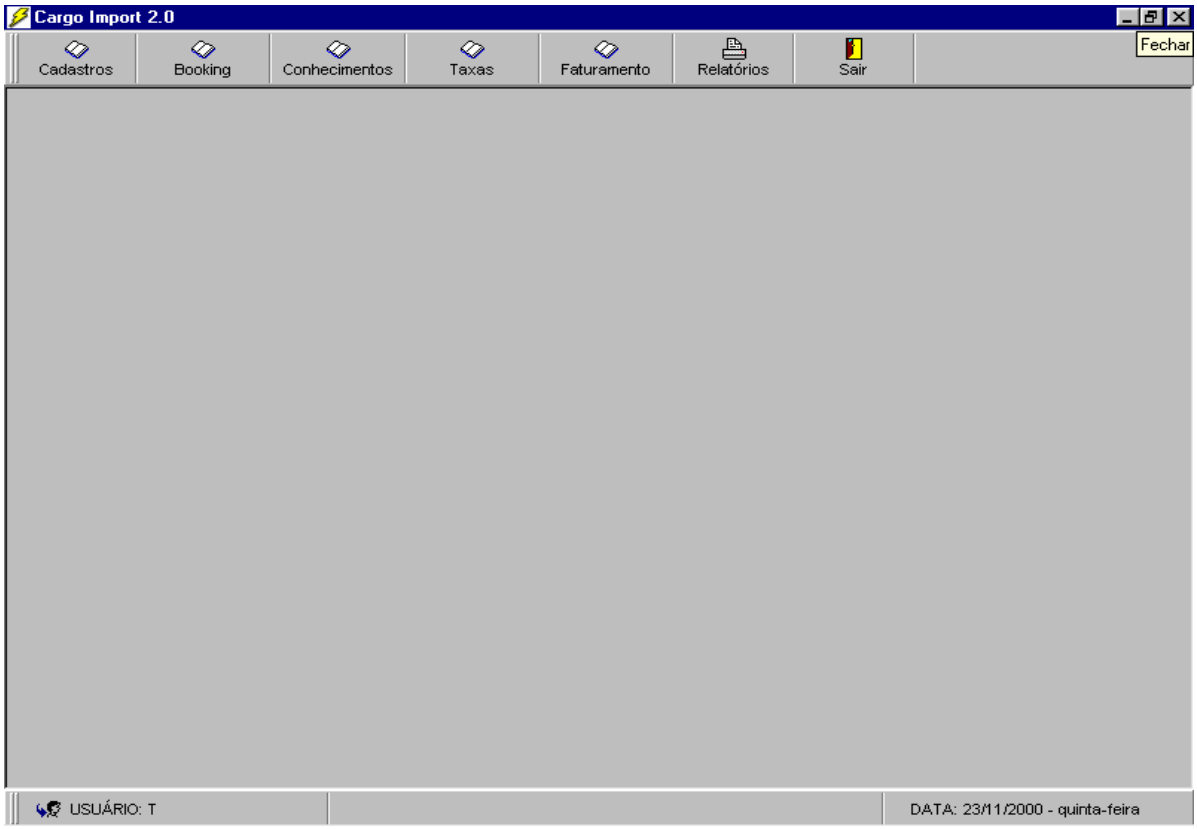
f) *as_getParams*: retorna o valor de um parâmetro;

g) *as_rowRequest*: retorna informação de um registro específico.

Raramente será preciso chamar estes métodos diretamente, pois os componentes do Delphi para serem usados nos lados cliente e servidor do aplicativo incorporam estas chamadas.

Quanto ao funcionamento sistemático do protótipo de sistema de importação (cliente), na visão do usuário, pode-se dizer que está distribuído em duas partes, a administrativa/operacional, onde todos os cadastros são efetuados, como clientes, companhias aéreas, aeroportos, agentes, taxas, criação do *booking* e conhecimento de carga, e a financeira. Na figura 17 é apresentada a tela de abertura do protótipo de sistema de importação.

FIGURA 17 - TELA PRINCIPAL DO PROTÓTIPO DE SISTEMA DE IMPORTAÇÃO



Todo o processo de importação começa pela cotação do frete internacional que poderá ser solicitada também através do site da empresa ou via telefone, mas na maioria das vezes, é feito através de correio eletrônico.

A partir do momento que o cliente aceita a cotação solicitada e autoriza o embarque da carga a ser transportada, é enviado à Blucargo Transportes a fatura comercial do processo de importação, cujas informações serão inseridas no documento *"booking"*, juntamente com detalhes da instrução de embarque passada para o agente embarcador da carga no exterior (figura 18). O *booking* pode ser considerado como sendo o documento que contém todas as informações necessárias para o embarque da carga.

Toda carga aérea ou marítima segue acompanhada por seu respectivo conhecimento aéreo AWB *(Air Way Bill).* A este devem seguir em anexo documentos específicos como notas fiscais, declarações e outros quando configurados casos especiais. Este documento tem validade durante todo o transporte a partir do momento da entrega para embarque até a entrega ao consignatário. Cobre várias finalidades como :

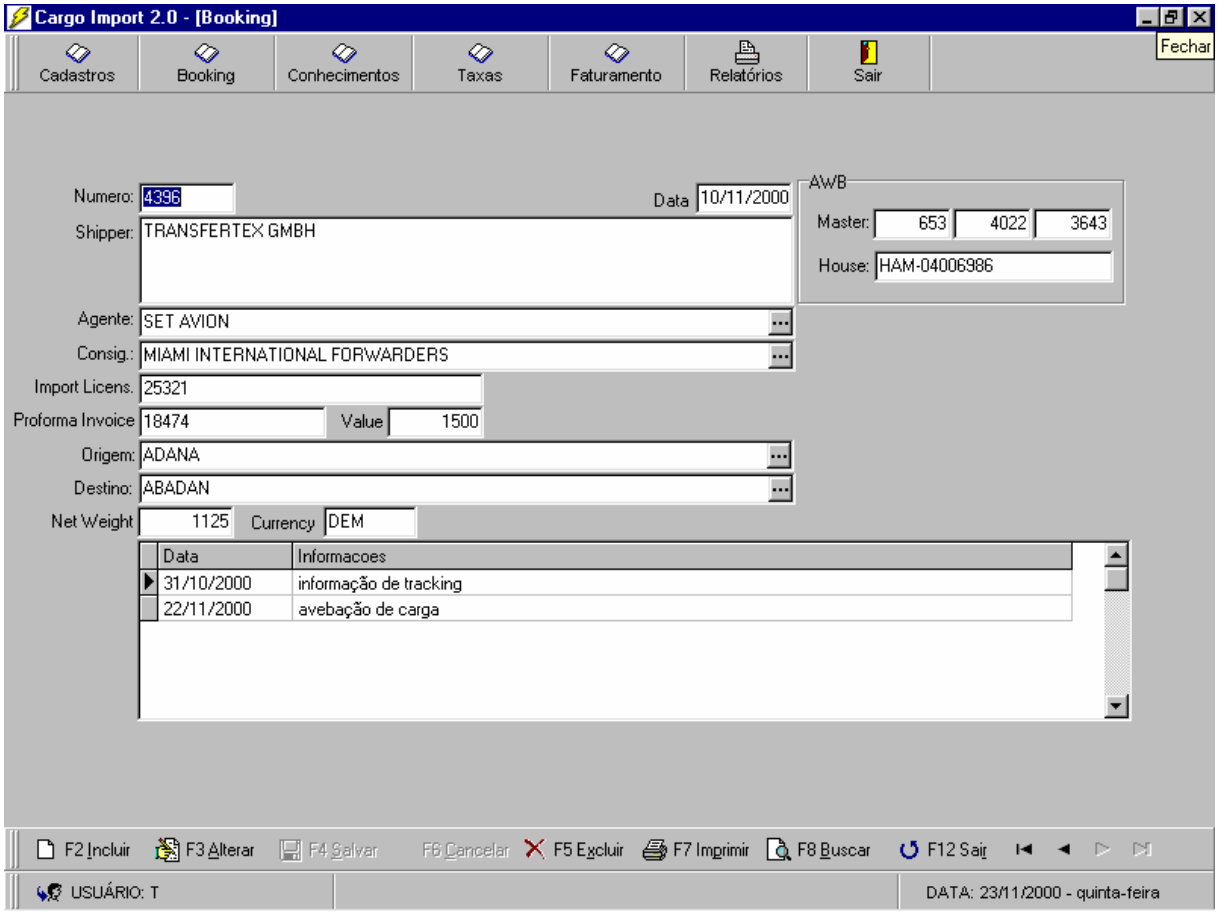
a) nota de frete;
b) certificado de seguro;
c) declaração alfandegária;
d) conhecimento para manuseio, despacho, entrega para embarque;
e) evidência documentária da conclusão do contrato de transporte.

Os embarques consolidados (agrupados), consistem no conjunto de várias cargas de vários importadores, com objetivo de reduzir o custo das tarifas aéreas ou marítimas, sendo que cada carga de um respectivo importador possui um AWB. Existem dois tipos de AWB no processo de embarque consolidada :

a) MAWB *(Master Air Way Bill) :* documento emitido pelo agente embarcador conforme número gerado pela companhia aérea ou marítima, que agrupa todas as informações contidas nos vários HAWB *(House Air Way Bill*) agregados. Este documento é consignado à Blucargo, declarando que o trânsito das cargas que acompanham os respectivos HAWB agregados são de sua responsabilidade;
b) HAWB : documento emitido pelo agente embarcador conforme número gerado pelo seu próprio sistema (de controle interno), e que presta informações detalhadas sobre a carga consignada ao importador. O documento é consignado

diretamente ao importador da carga, sendo de sua responsabilidade a nacionalização (liberação alfandegária) da carga.

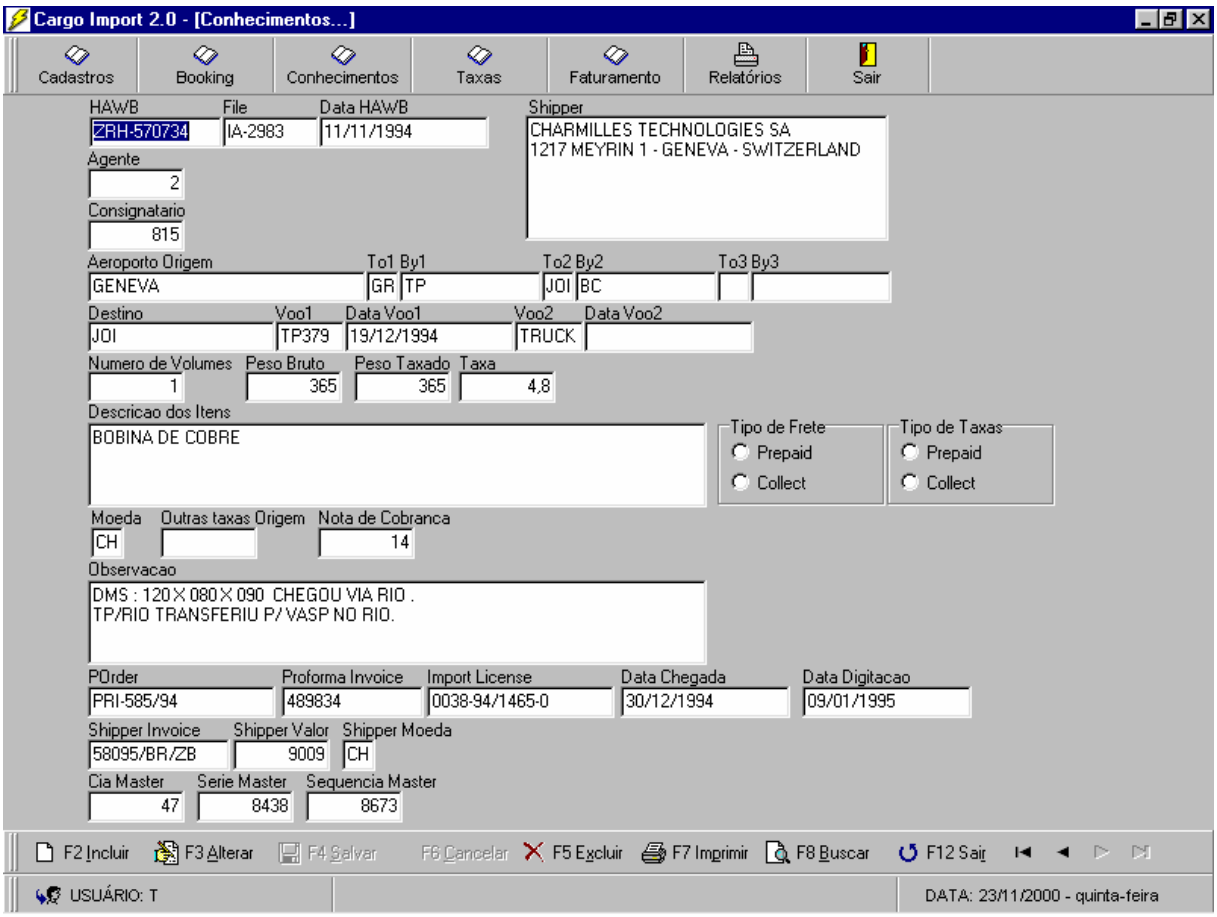
FIGURA 18 - BOOKING NO PROTÓTIPO DE SISTEMA DE IMPORTAÇÃO



Enfim, quando a carga importada chegar ao país de destino, a mercadoria recebida do exterior deve obrigatoriamente passar pela fiscalização aduaneira que é realizada pela Receita Federal recebendo assim, tratamento de carga importada, procedimento feito de acordo com a legislação vigente. Caberá à companhia aérea registrar no sistema da Receita Federal (MANTRA – Manifesto e Trânsito) a chegada da mercadoria, providenciar o envio da mesma para o armazém alfandegado da Infraero (Empresa Brasileira de Infra-Estrutura Aeroportuária) e a entrega da documentação para o cliente. Quando tratar-se de frete à pagar *(collect)*, a Blucargo Transportes será responsável pela cobrança do frete e das respectivas taxas inclusas no conhecimento aéreo relativas a essa modalidade de pagamento. Compete ao cliente, de posse dos documentos, a retirada da carga junto à alfândega e o pagamento de possíveis taxas e/ou impostos devidos, por exemplo taxa de armazenagem, imposto de importação e outros, no armazém alfandegado.

Assim que todo este processo estiver concluído, a mercadoria é despachada à cidade destino e então é emitido o conhecimento da carga, ou seja, o conjunto de todas as informações da carga que foi importada. O conhecimento é o mais importante documento de carga emitido pelo transportador ou agente de carga. É o equivalente ao termo de nota de consignação aérea, significando um contrato entre o embarcador e o transportador, para o transporte de bens sobre uma rota da transportadora, ou melhor, é aquele que agrega todos os dados do *booking*, clientes, companhias aéreas, informações da carga, como peso, volume, valor do agenciamento, valor do frete e etc (figura 19).

FIGURA 19 – FORMULÁRIO DO CONHECIMENTO NO PROTÓTIPO

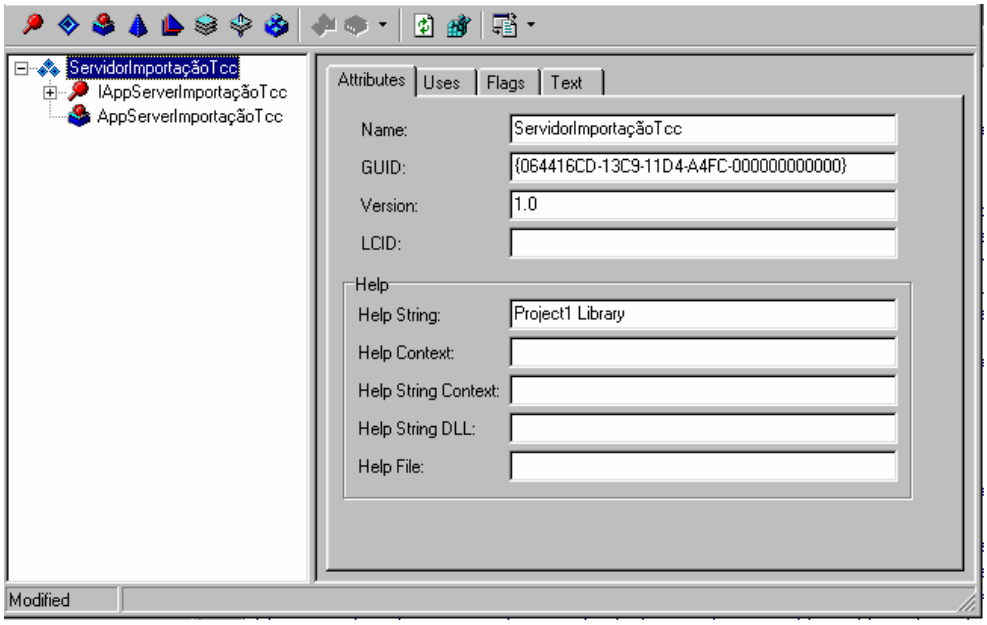


A segunda parte do protótipo de sistema de importação é a financeira, onde todo o processo relacionado a faturamento e controle financeiro é controlado, como emissão de recibos aos cliente e agentes, duplicatas, contrato de câmbio e a emissão das faturas. Como se trata de um protótipo, não houve tempo hábil para desenvolvimento da segunda parte do sistema.

5.3.2 CAMADA DE REGRAS DE NEGÓCIOS

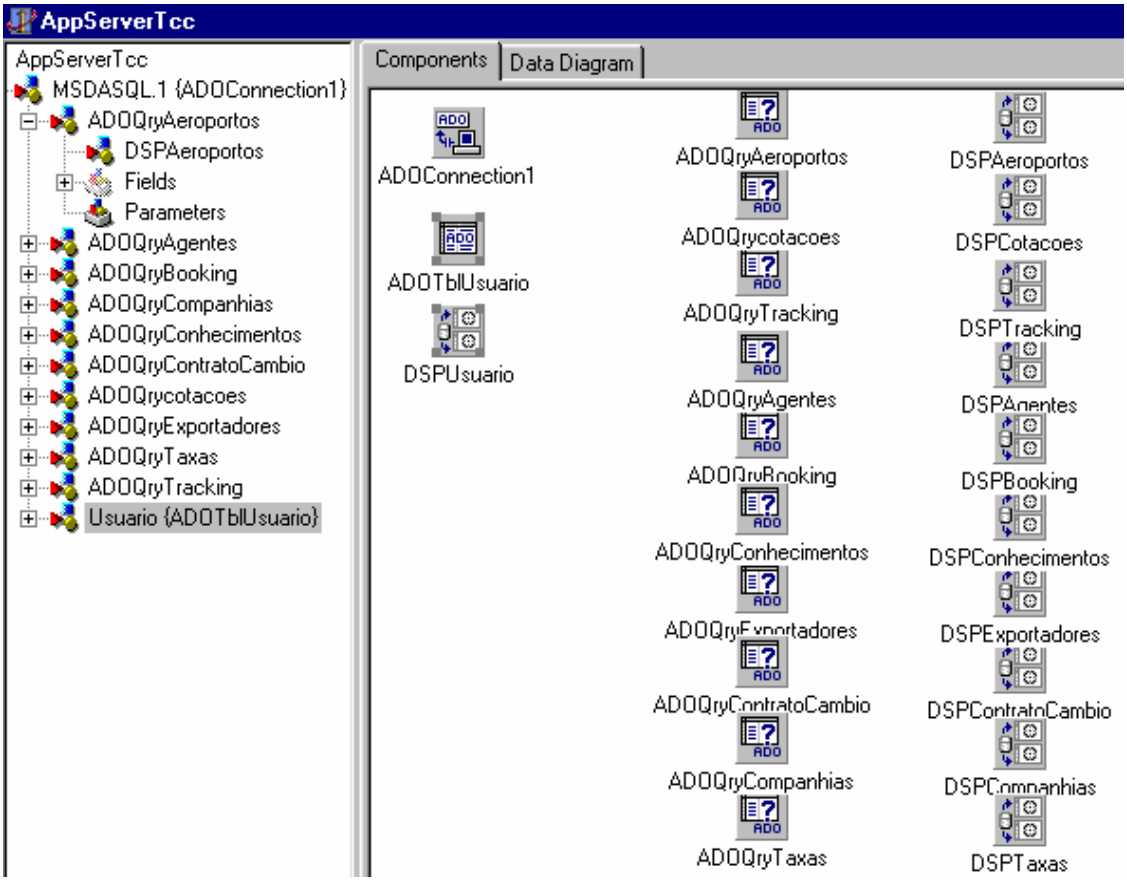
Para a implementação da camada de regras de negócios do protótipo de sistema de importação, foi criado um novo aplicativo no Delphi 5.0. Através do *Remote Data Module Wizard*, disponível no ambiente, deve ser informado o nome da classe do servidor, neste caso denominada *"AppServerImportaçãoTcc"*. Então, o Delphi coloca um módulo de dados no programa, que terá propriedades e eventos normais, mas a sua classe será definida na biblioteca de tipos (*Type Library*), que é criada automaticamente. É nesta *Type Library* que serão definidos todos os métodos e propriedades utilizadas no nível intermediário (figura 20).

FIGURA 20 - O EDITOR TYPE LIBRARY



A figura 21 demonstra o módulo de dados do ServidorImportaçãoTcc, onde estão os componentes que fazem os acessos aos dados do sistema.

FIGURA 21 - MÓDULO DE DADOS REMOTO DO SERVIDOR

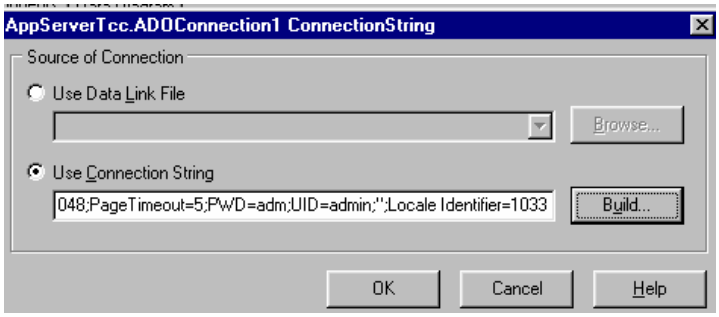


No protótipo de sistema de importação os componentes de acesso aos dados no lado do cliente e no lado servidor, comunicam-se via DCOM, e estão ligados aos dados através de componentes ADO.

No lado do servidor, componentes *TADOQuery e TADOTable*, estão ligados às tabelas de dados através de uma conexão ADO. A Conexão ADO (feita através do componente *TADOConnection*, pois cada componente ADOTable e *ADOQuery* está ligado ao *ADOConnection* através da propriedade *Connection*) encapsula o objeto *ADOConnection*. Uma conexão ADO pode prover múltiplos conjuntos de dados (*ADODataSet*) e comandos através de suas propriedades. A conexão permite ao programador o total controle dos atributos e condições sobre a base de dados conectada.

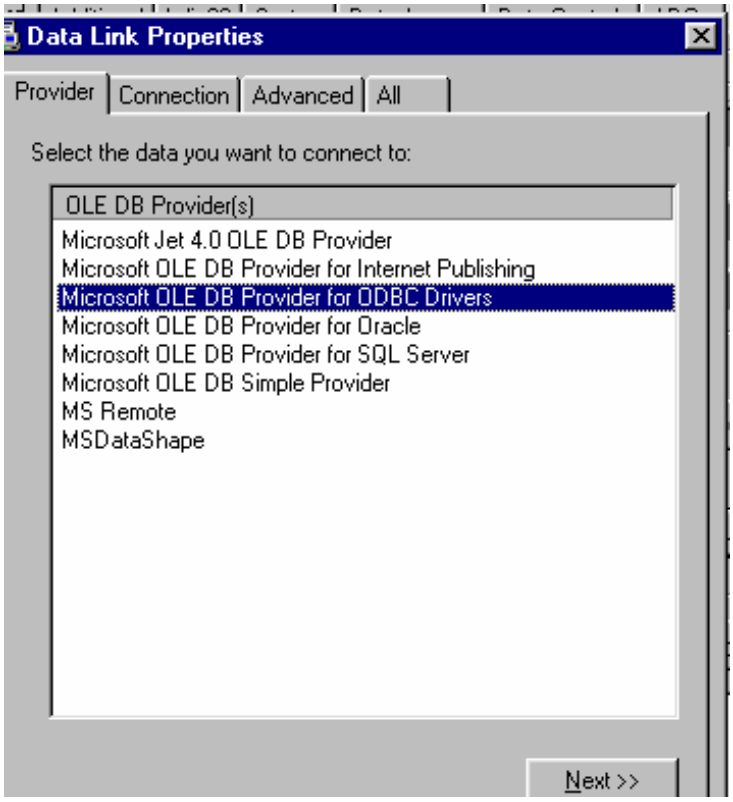
O componente *TADOConnection*, tem uma propriedade chamada *ConnectionString*, a qual deve ser configurada de acordo com a fonte de dados utilizada. A figura 22 demonstra a janela para configuração do *ConnectionString.*

FIGURA 22 - CONFIGURAÇÃO DA CONEXÃO ADO



Na guia *provider*, deve-se selecionar o provedor apropriado para o tipo de dados que se deseja acessar. Nem todos os aplicativos permitem que um provedor seja especificado (figura 23).

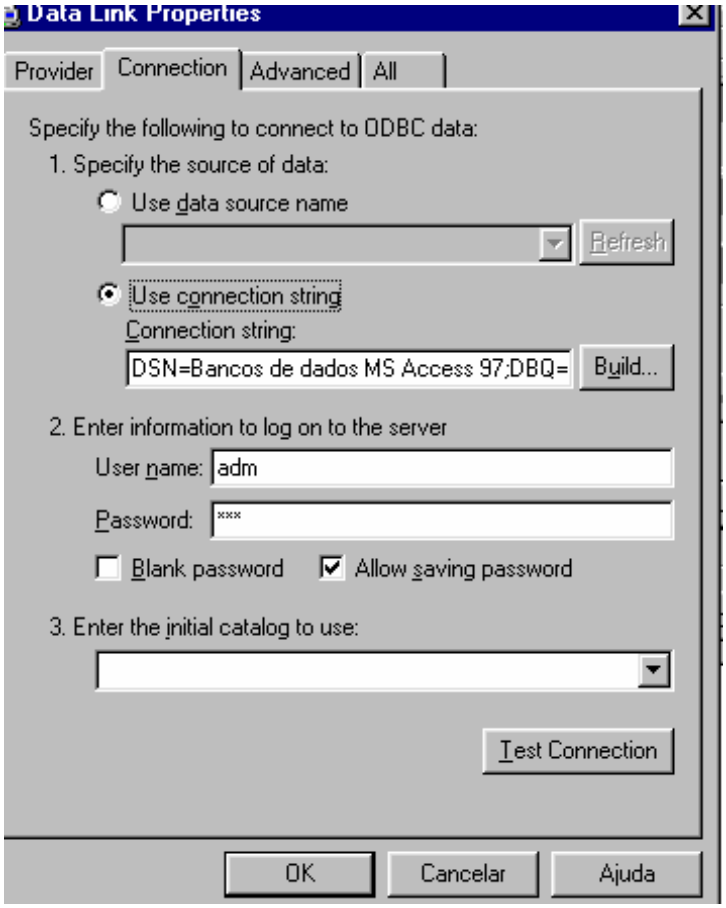
FIGURA 23 - CONFIGURAÇÃO DA GUIA PROVIDER



Na guia *Connection*, identifica-se o nome da fonte de dados (neste caso o nome do driver ODBC), e o usuário e senha de acesso (figura 24).

Na guia *Advanced*, são configuradas as permissões e na guia *All*, é demonstrado um resumo das configurações.

FIGURA 24 - CONFIGURAÇÃO DA GUIA CONNECTION



No protótipo de sistema de importação são utilizados componentes *TDataSetProvider* para prover os dados de uma tabela para o cliente, permitindo, ao mesmo, a alteração, inclusão e exclusão de dados que estão na camada de dados.

Para configurar o *TDataSetProvider*, basta informar qual é a fonte de dados associada a ela, através da propriedade *DataSet*. O cliente recebe um pacote de dados transmitido pelo *DataSetProvider*, utiliza este pacote como se fosse uma aplicação local, e depois aplica as alterações na base de dados.

As aplicações clientes podem acessar o *DataSetProvide*r através da interface *IAppServerImportacaoTCC*.

5.3.3 ACESSO A CAMADA DE DADOS VIA WEB

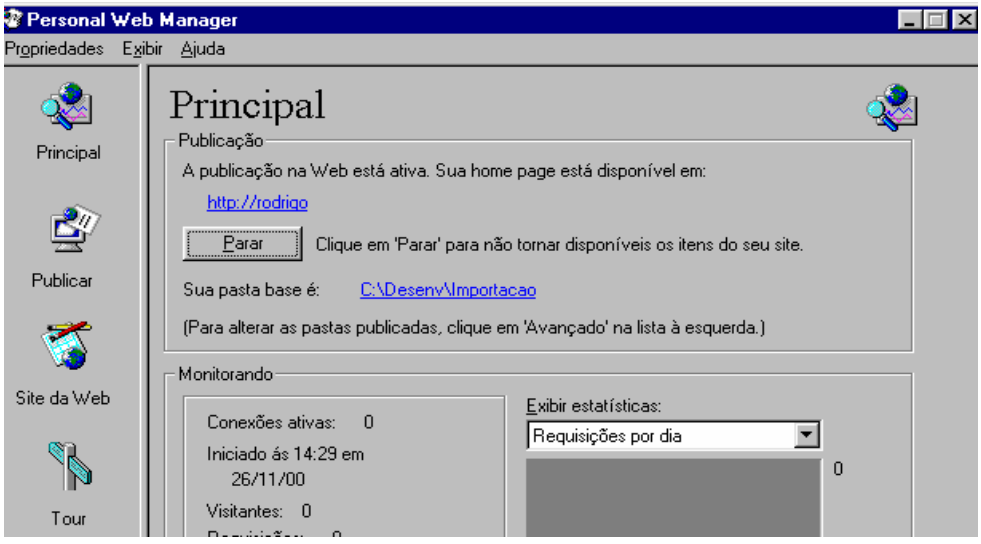
Para a implementação do acesso a camada de dados do protótipo de sistema de importação através do site da Blucargo Transportes, utilizou-se a tecnologia ASP, com recursos da ferramenta para edição de páginas *web DreamWeaver* 3.0 da *Macromedia*®, mas é possível implementar todo o código utilizando apenas o bloco de notas do Windows.

Uma das vantagens do ASP é a independência do *browser*, ou seja, ASP pode rodar páginas complexas no servidor (servidor Internet) e enviar somente os resultados para o cliente, e ainda permite visualizar, atualizar e adicionar informações através de SQL *(Structure Query Language)*. Como o servidor retorna somente o resultado HTML, o código fonte (lógica) fica preservado.

É necessário que o servidor que irá hospedar as páginas ASP utilize uma plataforma Windows e que possua o *Microsoft Internet Information Server* (IIS) e o *Personal Web Server* (PWS) instalado neste computador.

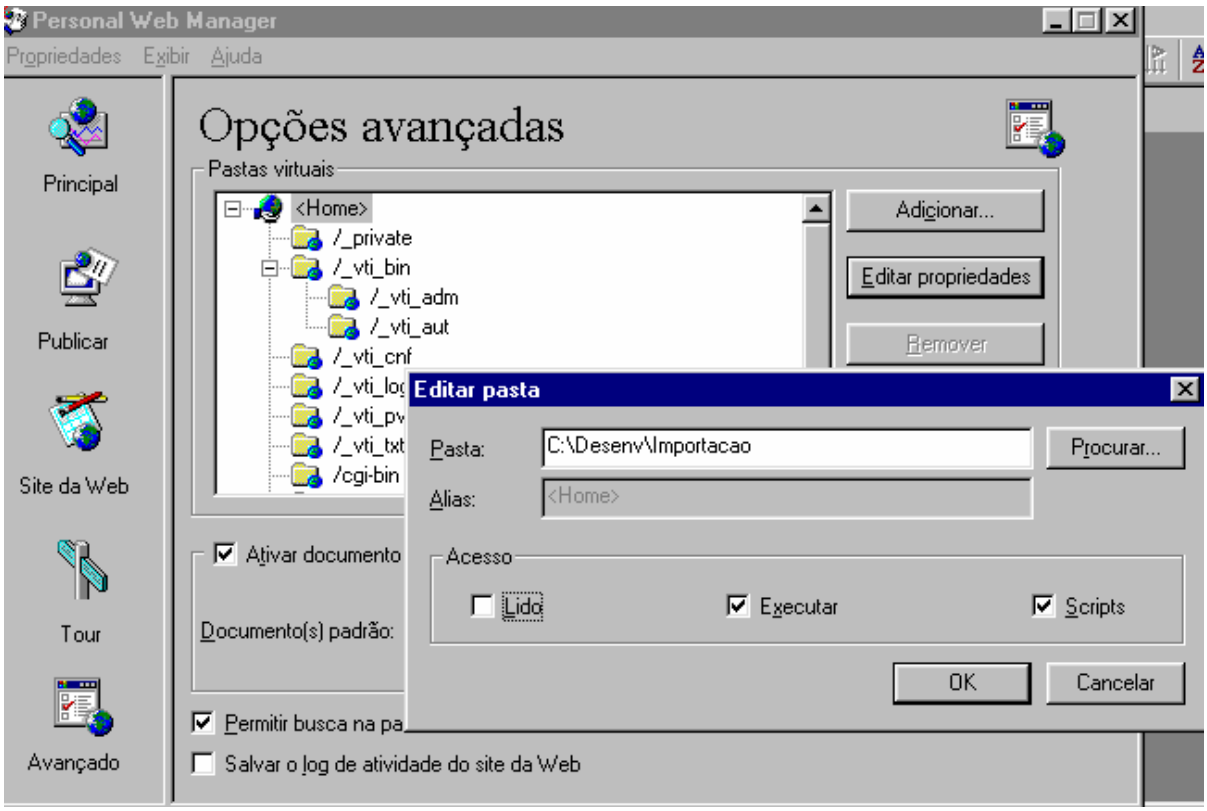
Para testes do código ASP no acesso à camada de dados no protótipo de sistema de importação, utilizou-se o *Personal Web Manager* (PWM), que é um gerenciador de recursos *web*, acoplado ao IIS e ao PWS, que possibilita emulação do servidor e cliente numa única máquina (figura 25).

FIGURA 25 - INTERFACE DO PERSONAL WEB MANAGER



O PWM permite que se defina um diretório para guardar as páginas ASP e inclua este diretório na lista do IIS/PWS, dando direito de "Execute/Execução". As páginas ASP só podem ser rodadas a partir de um diretório com o direito de "Execução" (figura 26).

FIGURA 26 - OPÇÕES AVANÇADAS DO PWM



Para exemplificar ASP no acesso a camada de dados no protótipo através do site da Blucargo Transportes foram implementados os cadastros de clientes e as cotações, onde toda pessoa física ou jurídica se cadastra automaticamente na base de dados do protótipo ou solicita uma cotação através de qualquer *browser* em qualquer lugar da Internet (figura 27).

FIGURA 27 - CADASTRO DE PESSOAS JURÍDICAS PELO SITE

Foi apresentado no quadro 20 o código ASP utilizado para o cadastro de clientes na base de dados do protótipo de sistema de importação.

Na classe *Connection* no código abaixo, tem-se o método *Execute* que permite executar uma estrutura SQL. Nesse caso, existe uma variável chamada SQL, que receberá o resultado do comando INSERT, inserindo na tabela de exportadores, os respectivos campos descritos.

QUADRO 20 - CÓDIGO ASP PARA CADASTRO DE CLIENTES NO SITE

```
<%
Dim con
Dim resultado
Dim Path
Dim SQL
Dim temp

Nome=request.form("Nome")
Endereco=request.form("Endereco")
```

```
Cidade=request.form("Cidade")
Estado=request.form("Estado")
Pais=request.form("Pais")
Telefone=request.form("Telefone")
FAX=request.form("FAX")
Cep=request.form("Cep")
email=request.form("email")
Inscricao=request.form("Inscricao")
CGC=request.form("CGC")
Contato=request.form("Contato")

chave = session("chave")
Set con=Server.CreateObject("ADODB.Connection")
path=Server.MapPath("Importacao.mdb")
con.open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source="& path
SQL="insert into Exportadores (Nome, Endereco, Cidade, Estado, Pais, Telefone, FAX, Cep,
email, Inscricao,CGC,Contato) values ('" & Nome &"','"& Endereco &"','"& Cidade &"',
'"& Estado &"','"& Pais &"','"& Telefone &"','"& FAX &"','"& Cep &"','"& email &"',
'"& Inscricao &"','"& CGC &"','"& Contato &"')"
Set resultado=con.Execute(SQL)
response.write "<br><br><center><b>"&"Sua Inclusão foi efetuada com sucesso !!!"&"
</center></b>"
%>
```

Quanto a solicitação de cotação através do site, acessando a opção “cotações importação”, o cliente terá a disposição um completo formulário a ser preenchido com todas as informações necessárias para o desenvolvimento da cotação, que será cadastrada automaticamente no protótipo (figura 28). Assim que a cotação foi concluída , as informações são retornadas ao cliente através de fax ou correio eletrônico.

FIGURA 28 - PEDIDO DE COTAÇÃO ATRAVÉS DO SITE

Foi apresentado no quadro 21 o código ASP utilizado para a solicitação de cotação através do site da Blucargo Transportes. O sistema emite um aviso ao usuário quando uma cotação nova é solicitada através do site.

QUADRO 21 - CÓDIGO ASP PARA SOLICITAÇÃO DE COTAÇÃO NO SITE

```
<%
Dim con
Dim resultado
Dim Path
Dim SQL
Dim SQL2
Dim temp

Nome=request.form("Nome")
Empresa=request.form("Empresa")
Assunto=request.form("Assunto")
email=request.form("email")
Fone=request.form("Fone")
Peso=request.form("Peso")
area=request.form("Area")
Volumes=request.form("Volumes")
Comprimento=request.form("Comprimento")
Largura=request.form("Largura")
Altura=request.form("Altura")
```

```
Origem=request.form("Origem")
Destino=request.form("Destino")

chave = session("chave")
Set con=Server.CreateObject("ADODB.Connection")
path=Server.MapPath("Importacao.mdb")
con.open "Provider=Microsoft.Jet.OLEDB.4.0;Data Source="& path
SQL2="select
Nome,Empresa,Assunto,email,Fone,Peso,Volumes,Comprimento,Largura,Altura,
[Altura]*[Largura]*[Comprimento]*[Volumes] as [Area], Origem, Destino
from Cotacoes"
SQL="insert into
Cotacoes(Nome,Empresa,Assunto,email,Fone,Peso,Volumes,Comprimento,Largura,Altura,Ar
ea,Origem,Destino) values ('" & Nome &"','"& Empresa &"','"& Assunto &"','"& email
&"','"& Fone &"','"& Peso &"','"& Volumes &"','"& Comprimento &"','"& Largura &"','"&
Altura &"','"& Altura*Largura*Comprimento*Volumes &"','"& Origem &"','"& Destino
&"')"
Set resultado=con.Execute(SQL)
response.write "<br><br><center><b>"&"Sua Cotação foi Solicitada com sucesso
!!!"&"</center></b>"
%>
```

Sendo assim, neste trabalho apresentou-se o desenvolvimento de um protótipo de um sistema de importação para a empresa Blucargo Transportes Nacionais e Internacionais Ltda, que servirá de base para o futuro desenvolvimento de um sistema permanente utilizando a ferramenta visual Delphi e as tecnologias ADO, COM/DCOM e ASP.

6 CONCLUSÃO

Devido a necessidades do mercado, muitos avanços estão ocorrendo na área de informática, principalmente na parte de orientação a objetos, onde estão sendo criados novos significados para a computação distribuída. As organizações estão cada vez mais se expandindo e necessitando de informações em qualquer hora e em qualquer lugar, não importando onde seja a fonte de dados.

A utilização de objetos distribuídos está cada vez mais presente em nosso dia a dia. O estudo da tecnologia ADO mostrou-se muito interessante para o aprimoramento das técnicas de acesso aos dados, uma vez que a tecnologia é fácil de utilizar, conforme sua própria documentação prega e o resultado obtido no protótipo satisfez as expectativas.

O Modelo DCOM proposto pela Microsoft, pelo que pode ser visto na medição de sua performance, sua resposta a invocação de métodos remotos é rápida. A maioria das características citadas na documentação de DCOM, como tolerância a falhas, independência de localização, compatibilidade com qualquer protocolo de rede e integração com qualquer plataforma de hardware puderam ser testadas ao decorrer do desenvolvimento do protótipo, obtendo êxito em todas elas.

Também é importante ressaltar a grande facilidade de programar usando DCOM. Sem grandes esforços é possível desenvolver uma pequena aplicação utilizando objetos distribuídos. A grande facilidade de programação fica ainda mais explícita quando da programação do lado cliente. De forma transparente, programa-se como se o objeto servidor estivesse na própria máquina local.

Quanto ao uso de ASP para acesso à base de dados do protótipo via *web*, pode-se destacar algumas vantagens como por exemplo a independência do *browser*, podendo executar páginas complexas no servidor e enviar somente os resultados para o cliente, e ainda a segurança do código fonte. Como o servidor retorna somente o resultado HTML, o código fonte (lógica) fica preservado.

O principal objetivo do trabalho, que era a especificação e implementação de um protótipo de um módulo de um sistema de importação foi atingido.

Referente aos objetivos secundários, que era a verificação da performance do uso de ADO, obteve-se sucesso na implementação deste conceito com ASP, o que agrega ao

protótipo de sistema de importação uma característica inovadora e atraente, e o que é mais importante, a utilidade desta implementação.

Sugere-se para trabalhos futuros, seria a implementação de tratamentos mais rigorosos quanto à segurança do sistema, pois ainda está muito vulnerável. Poderia ser feito um estudo sobre criptografia, implementando técnicas de segurança no protótipo de sistema de importação.

7 REFERÊNCIAS BIBLIOGRÁFICAS

[AMB1997] AMBLER, Scott W. **Análise e projeto orientados a objeto.** IBPI Press, 1997.

[CAN1999] CANTU, Marco. **Dominando o Delphi 5 : a bíblia**. São Paulo : Makron Books, 1999.

[CHR1999] CHRISTOPH, Wille, KOLLER Christian. **Aprenda em 24 horas Active Server Pages.** Rio de Janeiro : Campus, 1999.

[FUR1998] FURLAN, José Davi. **Modelagem de objetos através da UML – *the unified modeling language*.** São Paulo : Makron Books, 1998.

[MAR1999A] MARCORATTI, José Carlos. **ASP – ADO e banco de dados na Internet.** Florianópolis : Visual Books, 1999.

[MAR1999B] MARTINER, William, FALINO James, HERION David**. Building distributed applications with ADO**. New York : wiley Computer Publishing, 1999.

[MAR1999C] MARQUES, Alexandre de Moura. **Comércio exterior**. 1.ed. Porto Alegre : Síntese, 1999.

[MIC1999A] MICROSOFT CORPORATION. **DCOM Technical Overview.** http://msdn.microsoft.com/library/backgrnd/html/msdn_dcomtec.htm, Porto Alegre : UFRGS, 1999.

[MIC1999B] MICROSOFT CORPORATION. **DCOM Architecture.** http://www.msdn.microsoft.com/library/backgrnd/html/msdn_dcomarch.htm, Porto Alegre : UFRGS, 1999.

[MIC2000] MICROSOFT CORPORATION. **MSDN library.** www.msdn.microsoft.com. Porto Alegre : UFRGS, 2000.

[MIT2000] MITCHELL, Scott, ATKINSON James. **Aprenda em 21 dias ASP Active Server Pages 3.0.** Rio de Janeiro : Campus, 2000.

[RAJ1999] RAJ, Gopalan Suresh. **A Detailed Comparison of CORBA, DCOM and Java/RMI.** http://www.execpc.com/~gopalan/misc/compare.html. Porto Alegre: UFRGS, 1999.

[RAT1987] RATTI, Bruno. **Comércio internacional e câmbio.** São Paulo : Aduaneiras, 1987.

[SES1998] SESSIONS, ROGER**. COM and DCOM : Microsoft's vision for distributed objects.** Wiley Computer Publishing, 1998.

[SOS2000] SOSA, Roosevelt Baldomir. **A aduana e o comércio exterior**. 7.ed. São Paulo : Aduaneiras, 2000.