Universidade Federal de Santa Catarina

Sistema de alta disponibilidade para gerenciamento de pedidos e estoque

Mariell Schappo

Florianópolis – Santa Catarina

2012/2.

Universidade Federal de Santa Catarina

Departamento de Informática e Estatística

Curso de Ciências da Computação

Sistema de alta disponibilidade para gerenciamento de pedidos e estoque

Trabalho de conclusão de Curso apresentado como parte dos requisitos para obtenção do título de Bacharel, do curso de Ciências da Computação na Universidade Federal de Santa Catarina.

Florianópolis, 2012
2012/2

Mariell Schappo

Sistema de alta disponibilidade para gerenciamento de pedidos e estoque

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do grau de Bacharel em Ciências da Computação.

Orientador: Professor Doutor Frank Augusto Siqueira

Banca Examinadora:

Professora Doutora Patrícia Vilain

Professor Doutor Ronaldo dos Santos Mello

Dedico este trabalho à minha família, ao meu noivo, aos meus amigos e especialmente ao meu pai, por ter sido um grande apoiador de todas as minhas conquistas.

Resumo

Este projeto é uma aplicação de metodologias da área de computação distribuída na criação de um sistema altamente acoplável e expansível de criação de pedidos e atualização de reserva de estoque. Na maioria dos sistemas de venda e controle gerencial ocorre um grande número de operações relacionadas à efetivação de pedidos, enquanto outras áreas do sistema têm um fluxo moderado, como emissão de notas fiscais e relatórios gerenciais. Através de um sistema especializado, o grande fluxo de pedidos será gerenciado através de vários servidores que se comunicam para manter seus dados consistentes, sem perder a autonomia individual para realizar o pedido. Para cumprir estas especificações, cada servidor contará com um estoque local e um estoque global, e um servidor poderá efetuar instantaneamente um pedido se possuir estoque local. Caso contrário, terá que solicitar uma transferência de outro servidor. A comunicação entre os servidores se dá via troca de mensagens.

Palavras-Chave: ERP, Geração de pedidos, Controle de estoque distribuído, Vendas.

Abstract

This project is an application of methodologies in the area of distributed computing in the creation of a highly expandable and attachable system to create orders and update inventory reserves. In most sales and management systems, there is a large number of operations related to accomplishment of orders while other areas of the system has a moderate flow, such as issuing invoices and management reports. Through a specialized system, the flood of applications will be managed across multiple servers that communicate to keep data consistent, without losing their individual autonomy to make the request. To fulfill these specifications, each server will have a local inventory and a global inventory, and a server may do a request if it has local inventory, otherwise you will have to request a transfer from another server. Communication between the servers is done by message passing.

Keywords: ERP, orders generator, distributed inventory control, Sales

Sumário

Resumo 5

Abstract 6

Lista de Figuras 10

Lista de quadros 11

Lista de reduções 12

1. Introdução 13

1.1. Contextualização 13

1.2. Objetivos 13

1.3. Justificativa 14

1.4. Metodologia 15

1.5. Limitação do Escopo 15

1.6. Organização do Texto 16

2. Fundamentação Teórica 18

2.1. Sistemas Distribuídos 18

2.2. Web Services 18

2.3. Protocolo SOAP 19

2.4. Replicação de Dados 21

2.5. ERP 22

2.6. ORM 23

2.7. Trabalhos Relacionados 23

3. Tecnologia Utilizada 25

3.1. IIS (Internet Information Services) 25

3.2. .NET Framework 26

3.3. C# (C Sharp) 27

3.4. SQL Server 2008 27

3.5. ODBC 28

3.6. PhoneGap 28

3.7. HTML5 29

3.8. CSS 29

3.9. JavaScript 30

3.10. JQuery 30

3.11. Gendal 31

3.12. DBClassMapper 32

4. Planejamento e Implementação 34

4.1. Visão Geral do Sistema 34

4.2. Levantamento de Requisitos 35

4.3. Casos de Uso 36

4.4. Classes 39

4.4.1. Classes Objeto 39

4.4.2. Classes de Implementação 40
4.4.3. Classes de Persistência 41
4.5. Esquema do Banco de Dados 42
4.6. Estrutura do Sistema 43
4.7. Projetos de Implementação 45
4.7.1. Projeto Barsa 45
4.7.2. Projeto Dealer 46
4.7.3. Projeto Sherlock 48
4.7.4. Cliente Windows 49
4.7.5. Cliente Android 50
4.8. Funcionamento do Nodo 51
4.9. Sistemática do Controle de Estoque 53
4.10. Gravação do Pedido 56
5. Estudo de Caso 58
6. Conclusão 62
6.1. Trabalhos Futuros 63
7. Bibliografia 64

Lista de Figuras

Figura 1 - Protocolo SOAP 21

Figura 2 - Uso do JQuery 31

Figura 3 - DBClassMapper 32

Figura 4 - Comparação de Classes com o BD 33

Figura 5 - Esquema Geral do Sistema 35

Figura 6 - Casos de Uso 37

Figura 7- Diagrama das Classes Objeto 40

Figura 8 - Diagrama das Classes de Implementação 41

Figura 9 - Diagrama das Classes de Persistência 42

Figura 10 - Diagrama de Tabelas do Banco de Dados 43

Figura 11 - Diagrama Geral de Funcionamento do Sistema 44

Figura 12 - Interface do Cliente Windows 50

Figura 13 - Esquema do Nodo – As setas vermelhas indicam o fluxo de dados do nodo entre o nodo e o mundo exterior. A setas azuis representam as trocas de mensagens internas do nodo. 53

Figura 14 - Diagrama de sequência do controle de estoque no momento do pedido 54

Figura 15 - Distribuição dos Nodos 58

Lista de quadros

Quadro 1 - Plataformas e Respectivas Funcionalidades Suportadas no PhoneGap 29

Quadro 2 - Exemplo de Classe de Interação com Gendal 47

Quadro 3 - Quantidades e Estoques 60

Lista de reduções

ISAPI - Internet Server Application Programming Interface

HTTP - Hypertext Transfer Protocol

HTTPS - Hypertext Transfer Protocol Secure

FTP - File Transfer Protocol

FTPS - File Transfer Protocol Secure

SMTP - Simple Mail Transfer Protocol

NNTP - Network News Transfer Protocol

CLI - Common Language Infrastructure

iOS - Sistema operacional móvel da Apple Inc.

1. Introdução

1.1. Contextualização

Com o passar do tempo o setor de comércio está se modificando, concentrando-se cada vez mais em grandes empresas. Normalmente uma empresa, mesmo sedimentada, concentra seus dados em um único sistema de grande porte e geralmente em um único servidor com grande capacidade e com backups espalhados geograficamente.

O problema deste tipo desse modelo clássico é que mesmo com grande poder computacional do servidor, os clientes necessitam de uma boa estrutura de comunicação em rede para que o sistema tenha um bom desempenho para seus usuários finais, o que nem sempre é viável.

Este projeto é indicado principalmente para redes de lojas espalhadas geograficamente ou para vendas virtuais, onde o usuário do sistema é o próprio cliente final. Nestes dois casos, a complexidade de um sistema distribuído se justifica pela diminuição da necessidade de uma rede que suporte transações, mantendo uma velocidade aceitável 100% do tempo.

1.2. Objetivos

O objetivo deste projeto de conclusão de curso é desenvolver um módulo de pedidos em um ambiente distribuído que possa ser facilmente incrementado à medida que a demanda do sistema cresça, ou seja, o sistema deve balancear as solicitações de clientes através do uso de vários servidores de modo que todos eles forneçam dados

precisos em um tempo rápido, mesmo quando o sistema encontra-se muito distribuído geograficamente.

Existem diversas soluções em sistemas comerciais e industriais, porém o maior tráfego de informações encontra-se ao gerar pedidos, principalmente quando temos vendas on-line. Este projeto visa criar um módulo independente para a criação dos pedidos, que se comunica com o sistema principal através de Web Services. Desta forma a plataforma dos sistemas se torna uma decisão do usuário, ampliando ainda mais a aplicabilidade deste projeto.

1.3. Justificativa

Nos dias atuais, a maioria das soluções de softwares empresarias trabalha com o consolidado modelo cliente-servidor, principalmente pela herança de sistemas construídos no início dos anos 2000, conforme mostrado por Bergamaschi [4]. Na maioria dos casos esta abordagem funciona bem, pois o fluxo de dados é comportado de forma suficiente pela rede. Porém algumas empresas possuem pontos de vendas espalhados geograficamente, de modo que alguns nodos clientes são atendidos bem enquanto outros sofrem com a distância e a falta de infraestrutura principalmente na criação de pedido de vendas, onde o tempo de uma operação é algo muito importante e reflete diretamente na receita da empresa.

Algumas instituições contornam este problema com vários softwares ou vários bancos de dados, porém existe uma enorme desvantagem no ponto de vista gerencial. Este modelo acaba gerando inconsistências e burocratização da recuperação de informações.

Este projeto tem como alternativa separar apenas o módulo crítico de geração de pedidos, deixando os outros no modelo cliente-servidor, já que na prática não possuem um fluxo tão grande. A principal vantagem dessa separação do módulo crítico, é poder concentrar os esforços no ponto que mais trará benefícios para o sistema de modo geral.

1.4. Metodologia

O projeto se divide em parte teórica, apresentação da base teórica dos esquemas para a criação do sistema; e parte prática, em que este sistema proposto será de fato implementado de forma a contemplar os pré-requisitos apresentados na parte teórica.

A elaboração e construção desse projeto foram divididas nas seguintes etapas:

- Definição do escopo do projeto.
- Estudo do cenário atual do mercado.
- Embasamento teórico.
- Levantamento de Requisitos.
- Elaboração do sistema.
- Implementação do sistema.
- Execução e estudo de caso.

1.5. Limitação do Escopo

Este projeto não abrange todos os tópicos possíveis que o sistema de pedidos pode necessitar em sua aplicação comercial. A questão da segurança da informação é um item que não será tratado nesta monografia, pelo fato de pertencer a outro ramo de estudos, mas pode ser contemplada em projetos futuros.

Outro aspecto excluído do escopo é o controle de limitação de crediário para clientes, bem como o suporte a pedidos baseados em representantes (vendedores).

O estudo de questões relacionadas ao transporte das informações via rede também não será abordado neste projeto, apesar do funcionamento da rede ser de suma importância para o funcionamento do sistema.

Todo o mecanismo de pagamento não será alvo do projeto. Por se tratar de um ramo extenso de funcionalidades, além de depender diretamente do desenvolvimento de técnicas de segurança, será excluído do escopo.

Outra funcionalidade que não será abordada e nem colocada em prática neste projeto é o controle de filiais de venda e almoxarifados. Apesar deste projeto não contemplar filiais, ele foi construído de maneira a facilitar a inclusão desta funcionalidade.

1.6. Organização do Texto

No próximo capítulo serão abordados os aspectos teóricos que fazem parte do desenvolvimento do sistema. Os assuntos mais gerais serão tratados primeiramente para permitir um entrosamento com os conceitos abordados mais adiante.

O capítulo 3 apresenta um resumo dos aspectos gerais de cada tecnologia utilizada no projeto, além da justificativa de uso de cada uma delas.

O planejamento de como o projeto será implantado é abordado no capítulo 4, que mostra tanto os requisitos do sistema, quanto a forma de se colocá-los em prática para que o sistema se adeque da melhor forma possível às necessidades do projeto.

2. Fundamentação Teórica

2.1. Sistemas Distribuídos

As técnicas utilizadas neste projeto são baseadas em conceitos e métodos já conhecidos e utilizados em sistemas distribuídos, porém adaptados às necessidades deste. O módulo de pedidos depende de vários servidores trabalhando simultaneamente, de modo que a falha de um servidor não seja capaz de interferir na operabilidade do sistema de forma geral.

Segundo Coulouris [1] “Um sistema distribuído é aquele no qual os componentes localizados em computadores interligados em rede se comunicam e coordenam suas ações apenas passando mensagens”. O efeito deste tipo de sistema é uma maior complexidade na implementação do software, porém propicia vantagens em relação à disponibilidade e à agilidade do sistema, quando bem implementado, se comparado a um modelo centralizado de computação.

2.2. Web Services

A escolha da utilização de web services neste projeto deve-se ao fato de que esta é uma maneira de fornecer uma interface de serviço de forma muito abrangente, fazendo com que os clientes possam se comunicar com os servidores de forma simples e independente de plataforma e tecnologia [14]. Apesar de esta aplicação utilizar uma plataforma bem definida nos servidores, não existe uma obrigação em torno de tecnologia a ser aplicada nos clientes. Um cliente capaz de interagir com um protocolo

padrão não precisa conhecer os aspectos da plataforma do servidor gerando uma liberdade muito apreciada, principalmente no ambiente empresarial, foco deste projeto.

O sucesso dos web services é consequência da prova de eficácia de protocolos e técnicas simples que a disseminação da internet nos trouxe. A abrangência de um produto é fator decisivo para que haja uma utilização real deste e para que um software se torne abrangente ele deve ser suficientemente bom e proporcionalmente fácil de ser utilizado, quando estamos lidando com um ambiente exterior ao meio acadêmico.

Os web services surgiram com a necessidade que as aplicações da internet possuem de estabelecer e utilizar serviços de outras aplicações, pois o protocolo HTTP não foi elaborado com este intuito. Para solucionar esta questão, os web services podem processar troca de mensagens formatadas em XML, de forma a suportar passagem de parâmetros e retorno de funções. Para que o serviço possa ser corretamente utilizado, normalmente é criada uma descrição WSDL contendo os cabeçalhos e interfaces do serviço.

Utilizar web services neste contexto é uma maneira de abrir a solução para ser utilizada por clientes heterogêneos, adaptados a soluções locais de interface e usabilidade, deixando o núcleo do sistema livre para operar de modo a otimizar o processo.

2.3. Protocolo SOAP

Para a implementação dos web services deste sistema, é utilizado o protocolo SOAP, conhecido como *Simple Object Acces Protocol*. Mensagens SOAP são utilizadas principalmente baseadas em HTTP, porém o protocolo pode ser também em conjunto com outros protocolos de transporte. De modo geral, este protocolo determina de que

forma o XML deve ser estruturado para que transporte o conteúdo das mensagens com um padrão reconhecido pelo destinatário independente da linguagem em que o software foi construído.

As mensagens SOAP são estruturadas na forma de um envelope. Dentro do envelope está o cabeçalho, que pode transportar informações adicionais de contexto; e o corpo, onde se encontra a mensagem XML que deve ser lida pelo destinatário. As informações contidas no cabeçalho dependem muito da forma como um serviço será utilizado, pois podem ser usados por qualquer serviço de middleware de mais alto nível, podendo conter vários tipos de informações úteis para ele.

As informações de um serviço publicado ficam disponíveis em um arquivo WSDL *(Web Services Description Language*), registrados em um serviço de diretório chamado UDDI (*Universal Description, Discovery and Integration*). Quando uma aplicação deseja acessar um Web Service, ela procura por sua especificação (WSDL) e assim que possuir as informações necessárias, poderá se comunicar diretamente com o servidor do serviço. Este processo é mostrado na figura 1.

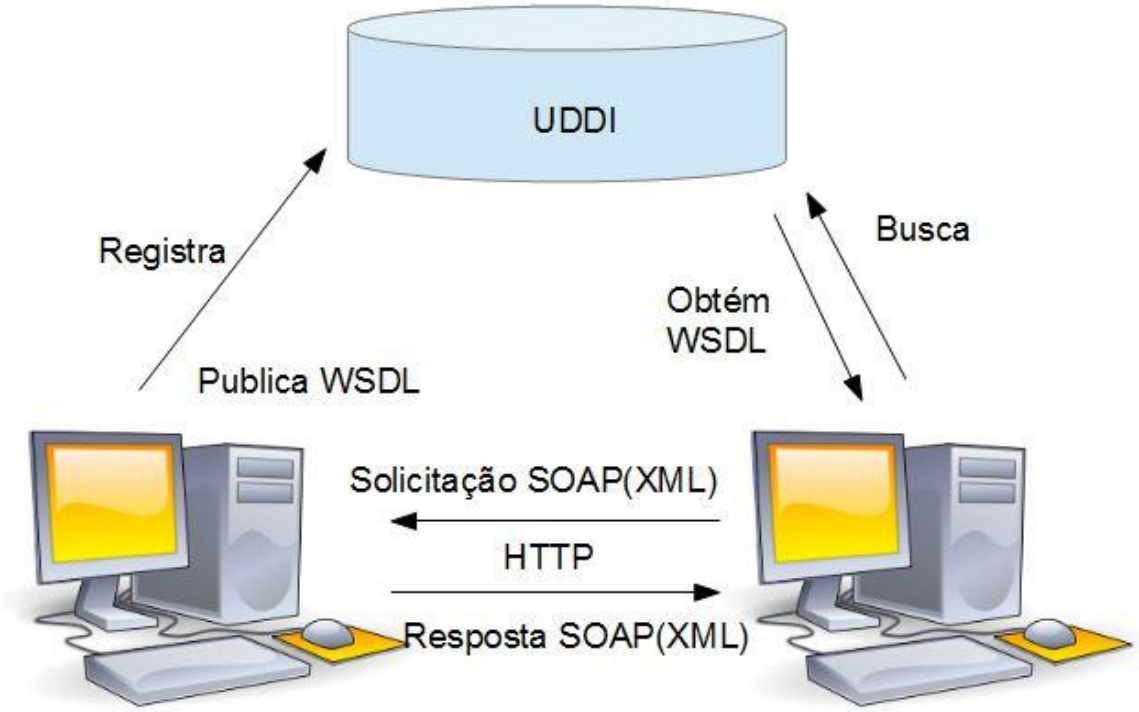

Figura 1 - Protocolo SOAP

2.4. Replicação de Dados

Para que os requisitos propostos por este projeto sejam atendidos é necessária uma replicação de dados entre os servidores para garantir a disponibilidade do sistema e a qualidade do serviço. Basicamente, a replicação de dados pode ser dividida em dois modelos: ativo e passivo [1].

O modelo passivo, também conhecido como backup primário, é caracterizado pela replicação feita através de um único gerenciador de réplica primário e um ou mais gerenciadores de réplica secundários. Quem trata toda a comunicação com o meio

externo é o gerenciador primário, e este se comunica com os outros gerenciadores. Quando o gerenciador primário cai, um secundário será encarregado de tomar o lugar do primário.

Na replicação ativa os gerenciadores de réplicas têm papéis equivalentes, e a comunicação com o *front end* é feita sem distinção entre os gerenciadores. Todos eles processam a requisição, mandada via *multicast* de forma idêntica e cabe ao *front end* analisar a resposta de cada servidor.

Neste projeto a replicação de dados será implementada no modelo ativo, com algumas adaptações que serão entendidas nos capítulos a seguir, onde o conceito e a estrutura geral deste sistema serão abordados.

2.5. ERP

ERP (Enterprise Resource Planning), conhecido no Brasil também como Sistema Integrado de Gestão Empresarial, é um modelo de solução empresarial que visa tratar o processo total de uma empresa dentro de um único sistema, ou seja, um sistema que centraliza todas as áreas da empresa.

A utilização de ERP's não é uma novidade no mercado [4], porém os já existentes tem uma tendência de se manter por muito tempo, até porque a construção de um ERP completo demanda muito tempo e dinheiro, além dos custos de implantação e treinamento de um novo sistema que vai modificar a rotina de trabalho de uma empresa inteira.

Os fatores mais importantes dentro de um ERP são a confiabilidade e segurança dos dados. Isso se deve ao valor que as informações possuem em um ambiente empresarial, onde dados podem ser a chave do sucesso ou do fracasso de instituição.

Este projeto implementa um sistema que se interliga com ERP's através de um web service. O objetivo é implementar o módulo de pedidos mais eficiente que os que a maioria dos ERP's possuem. Existe a possibilidade de integrar o sistema a um outro sistema mais simples que um ERP, porém dificilmente uma grande empresa terá um sistema de gestão pouco complexo.

2.6. ORM

Object-Relational Mapping é uma técnica utilizada para melhorar integração entre a programação orientada a objetos e a camada de acesso a banco de dados relacionais [5]. As ferramentas ORM costumam agilizar o processo de mapeamento do banco de dados em objetos. Ao utilizar ORM o programador não precisa se preocupar com comandos SQL, basta manipular os objetos. Além disso, algumas ferramentas estão preparadas para fazer automaticamente a conversão dados relacionais – objetos e vice versa, aumentando muito a produtividade do programador e minimizando a possibilidade de erros humanos nesta etapa do projeto.

2.7. Trabalhos Relacionados

Atualmente existem muitas soluções em software para a criação de pedidos, porém a grande maioria utiliza o modelo cliente-servidor clássico. Porém a utilização de Web Services para o desenvolvimento de soluções baseadas em múltiplos computadores já

foi empregada em alguns projetos entre eles o de autoria de Sabrina da Silva Leandro, projeto de conclusão do curso de Sistemas da Informação, no primeiro semestre de 2005, na Universidade Federal de Santa Catarina [14]. Este trabalho visa à interligação de sistemas heterogêneos através de web service a fim de melhorar o desempenho através de balanceamento de carga. O problema abordado no projeto descrito é solucionado de maneira semelhante a este projeto, sendo a principal distinção entre os dois, o fato de este projeto ter uma aplicação bem focada a um tipo de problema específico, a geração de pedidos.

A principal diferença entre estes dois projetos está na generalização na maneira como é exposto o problema e a solução no projeto de Sabrina da Silva Leandro, que diferentemente deste, abordou o tema com uma aplicabilidade geral, e não focando eu uma solução específica.

3. Tecnologias Utilizadas

O sistema foi construído na plataforma de servidor web Microsoft IIS, disponível para o sistema operacional Windows. A escolha da plataforma Windows se deve ao fato do ORM utilizado (Gendal) ter sido desenvolvido para uso desta. A escolha deste ORM será abordada com detalhes na seção 3.3. O framework escolhido para implementar este projeto foi o .NET 4.0, pois além da familiaridade, este framework possui todas as ferramentas necessárias para a execução do projeto, além de ser um ambiente de programação amigável, pois o projeto é focado na base teórica e nas técnicas implementadas e não em aspectos específicos de cada linguagem de programação.

3.1. IIS (Internet Information Services)

O IIS é uma ferramenta inclusa em algumas versões do Windows para hospedagem de serviços web. A versão mais recente até então (IIS 7.5) suporta HTTP, HTTPS, FTP, FTPS, SMTP e NNTP.

O IIS é encontrado nas versões servidores do Windows e em algumas versões “Professional” e “Ultimate”. Pelo fato do IIS ser fortemente integrado ao sistema operacional, ele é relativamente fácil de administrar. Sua maior desvantagem é o fato de rodar apenas em plataforma Windows, enquanto grande parte dos servidores web roda em outros sistemas operacionais.

Segundo o site oficial do IIS [2], as razões para desenvolver um projeto para a plataforma IIS são:

- Deixar aplicações web mais poderosas através das funcionalidades fornecidas como esquemas de autenticação, monitoramento e criação de logs, filtros de segurança, balanceamento de carga, redirecionamento de conteúdo e gerenciamento de estados.

- Facilidade no desenvolvimento de aplicações com o novo modelo de extensibilidade C++ resolve grande parte dos problemas do desenvolvimento ISAPI.

- Módulos ASP.NET podem prestar serviços de maneira uniforme a ASP, CGI, arquivos estáticos e outros tipos de conteúdo, e pode estender completamente o servidor sem a limitação presente em versões anteriores do IIS.

3.2. .NET Framework

O framework .NET surgiu como iniciativa da Microsoft para criar uma plataforma de desenvolvimento integrada, para que o mesmo código (escrito em inúmeras linguagens) possa ser facilmente compilado para diversas plataformas. O funcionamento destes requisitos deve-se ao CLR (*Commom Language Runtime*), capaz de executar diferentes linguagens de programação que utilizam um único framework.

Os programas feitos em .NET são duplamente compilados, como é o caso da linguagem Java, gerando um *bytecode* como código intermediário em uma linguagem conhecida como MSIL (*Microsoft Intermediate Language*). Na hora da execução é empregado um compilador *Just In Time* para a compilação do código intermediário. O

overhead da compilação JIT só ocorre na primeira vez em que o programa é executado, pois nas próximas vezes será usada a compilação realizada anteriormente. Existe também a possibilidade de gerar o código totalmente compilado para evitar o atraso da primeira execução.

3.3. C# (C Sharp)

A definição superficial para a linguagem C# em Robinson [10] é a de uma nova linguagem desenvolvida a fim de trabalhar em conjunto com o framework .NET e tirar proveito de todo o ambiente de desenvolvimento criado juntamente com o entendimento dos desenvolvedores sobre orientação a objetos ao longo de 20 anos. Esta linguagem tem como características o fato de ser fortemente tipada, orientada a objetos e simples.

A sintaxe de C# foi originalmente inspirada na linguagem C++, porém possui influências de Object Pascal e Java. O início do desenvolvimento desta linguagem ocorreu em 1999 e em 2003 tornou-se um padrão ISO. Os programas escritos em C# passam por uma compilação que os transformam em linguagem intermediária (IL) e o armazenam em disco. Na hora da execução, o CLR (*Common Language Runtime*) é quem irá transformar a IL em instruções de máquina, ou seja, uma compilação *Just-In-Time* (JIT) [11].

3.4. SQL Server 2008

SQL Server é um SGBD (Sistema de Gerenciamento de Banco de Dados) relacional criado em 1988 pela parceria entre as empresas Microsoft e Sybase, visando

principalmente o mercado corporativo. Segundo o site oficial do SQL Server [12], o SQL Server 2012 oferece maior segurança e maior produtividade de desenvolvimento se comparado com um de seus principais concorrentes, o Oracle.

3.5. ODBC

ODBC é uma sigla para *Open Data Base Connectivity*, ou seja, é um conjunto de protocolos para unificar o modo de acesso a dados independentemente da tecnologia empregada no banco de dados e da linguagem do programa que deseja ter acesso aos dados, desde que exista suporte ao ODBC.

3.6. PhoneGap

PhoneGap é um framework de código aberto para a construção de aplicativos móveis multi-plataforma, utilizando HTML5, JavaScript e CSS [6]. Este framework foi utilizado neste projeto na implementação do cliente Android, que será abordado em detalhes no capítulo 4. Uma das vantagens em construir aplicações com o phoneGap é que não é necessário aprender uma ferramenta de desenvolvimento para cada plataforma de dispositivos móveis. Tendo conhecimento de HTML, JavaScript e CSS, que são tecnologias extremamente difundidas e amplamente utilizadas, pode-se construir aplicações em todas as plataformas suportadas pelo PhoneGap, respeitando as limitações de funcionalidades demonstradas no quadro a seguir.

Funcionalidade	**iPhone/ iPhone 3G**	**iPhone 3GS +**	**Android**	**Windows Phone 7**	**BlackBerry 5.x**	**BlackBerry 6.0+**	**Sybian**	**Bada**	**Web OS**
Acelerômetro	X	X	X	X	X	X	X	X	X

Câmera	X	X	X	X	X	X	X	X	X
Compasso		X	X	X				X	X
Contatos	X	X	X	X	X	X	X	X	
Arquivos	X	X	X	X	X	X			
GeoLocalização	X	X	X	X	X	X	X	X	X
Midia	X	X	X	X					
Rede	X	X	X	X	X	X	X	X	X
Notificações	X	X	X	X	X	X	X	X	X
Armazenamento	X	X	X	X	X	X	X		X

Tabela 1 - Plataformas e Respectivas Funcionalidades Suportadas no PhoneGap

Outra vantagem é a reutilização de um mesmo projeto para vários tipos de aparelhos e plataformas, visto que a diversidade de sistemas operacionais móveis, assim como a incompatibilidade entre eles, é muito grande.

3.7. HTML5

HTML5 é a quinta versão da linguagem HTML, feita para codificar a estruturação e a apresentação de páginas web. Com o passar dos tempos, a linguagem HTML acabou sendo usada em conjunto com outras linguagens na elaboração de aplicações cada vez mais complexas. O HTML5 surgiu para melhorar a integração multimídia mais recente. Neste projeto, o HTML5 é usado na criação de um aplicativo para aparelhos móveis que será abordado na seção 4.7.5.

3.8. CSS

A necessidade de separar a estrutura de uma aplicação web de seus padrões gráficos tornou o CSS muito popular. A sigla CSS vem de *Cascading Style Sheets,* ou seja, o

desenvolvedor constrói uma folha de estilo em CSS que será vinculada a todas as páginas que necessitam usar o mesmo estilo gráfico.

3.9. JavaScript

JavaScript é uma linguagem de script, implementada pela maioria dos navegadores existentes desde a década de 90. Os scripts feitos em JavaScript são executados no lado cliente da aplicação web, o que torna esta uma linguagem muito eficiente para funcionalidades que devem ser rápidas e não necessitam de validação do lado do servidor. A vantagem que este tipo de programação proporciona é a não ocupação do servidor em tarefas desnecessárias e a rapidez de resposta que o usuário final possui. Neste projeto, JavaScript é utilizado para criar as animações e as interações com usuário existentes no cliente Android.

3.10. JQuery

JQuery é uma biblioteca criada para facilitar o desenvolvimento em JavaScript. A função do JQuery é facilitar as chamadas mais comuns em JavaScript e com isso aumentar a produtividade do programador. A Figura 2 mostra um exemplo de código feito utilizando o JQuery.

```
1.  <script type="text/javascript">
2.
3.  $(document).ready(function(){
4.
5.  $("#botao").click(function() {
6.
7.  alert('Olá mundo');
8.
9.  });
10.
11. });
12.
13. </script>
14.
```

Figura 2 - Uso do JQuery

A linha de código "*$("#botao").click(…)",* em JavaScript puro seria escrita: "*document.getElementById("botao").click*". Como podemos observar, o uso do JQuery faz com que o código fique menor, e consequentemente de melhor entendimento para os desenvolvedores.

3.11. Gendal

O Gendal é um ORM (Object Relational-mapping) desenvolvido como trabalho de conclusão de curso do aluno Diego Magno da Silva [5]. A utilização de um ORM é importante para facilitar o desenvolvimento de um sistema orientado a objetos, como é o caso deste, de forma a separar bem a camada de acesso a dados e utilizá-la em um nível mais alto, com a manipulação de objetos.

A importância do uso de uma ferramenta ORM deve-se ao fato de que um dos objetivos deste projeto é a elaboração de um sistema facilmente acoplável e que possa ser ampliado à medida que a demanda no sistema cresça. Existe no mercado uma imensa variedade de SGBDs, cada um com capacidade, características, limitações e

preços diferentes. Deixar o sistema fortemente acoplado a um SGBD específico poderia aumentar muito o custo de implantação para uma solução de pequeno porte ou, por outro lado, inviabilizar o armazenamento de dados em grandes quantidades com segurança e rapidez.

A escolha do Gendal aconteceu principalmente pela escassez de ferramentas ORM no mercado, pela dificuldade de utilização destas e pela limitação de funcionalidades.

3.12. DBClassMapper

Segundo Silva [5], o DBClassMapper é uma ferramenta de auxílio ao ORM Gendal. Ele é a ferramenta que auxilia visualmente a criação de classes e propriedades mapeadas diretamente no banco.

Primeiramente o DBClassMapper lê o banco de dados e proporciona ao usuário uma interface gráfica para montar o mapeamento das tabelas do banco para um conjunto de classes já configuradas conforme os padrões do GenDal. As figuras a seguir mostram o DBClassMapper em uso.

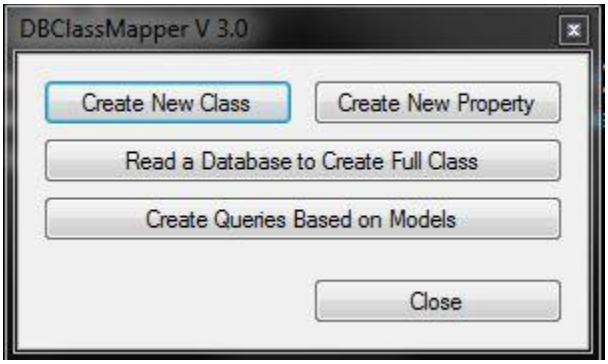

Figura 3 - DBClassMapper

A Figura 3 mostra a tela inicial do DBClassMapper, com as opções disponíveis para a criação das classes e propriedades. Neste trabalho foi utilizada a opção "*Read a Database to Create Full Class*" para o DBClassMapper ler a base de dados e gerar automaticamente as classes necessárias para o Gendal.

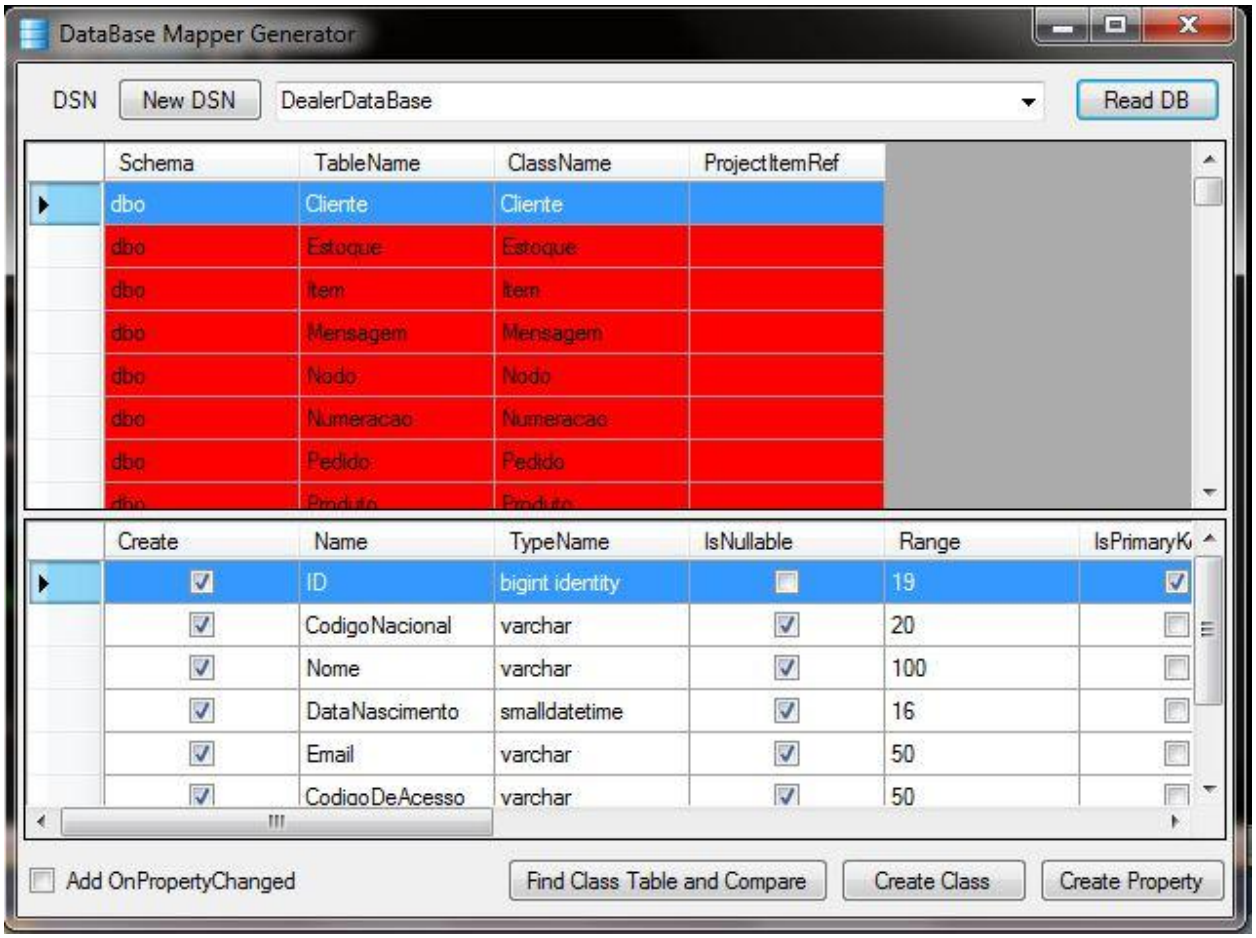

Figura 4 - Comparação de Classes com o BD

A Figura 4 mostra a interface de criação das classes fazendo o comparativo das classes existentes com as tabelas do banco de dados.

4. Planejamento e Implementação

4.1. Visão Geral do Sistema

O sistema desenvolvido neste projeto provê a criação de pedidos de compra através de uma estrutura combinada de vários nodos, em contrapartida ao modelo clássico cliente-servidor. O sistema permite comunicação do cliente e do ERP com qualquer nodo pertencente ao conjunto de nodos operantes. O ERP alimenta o sistema com dados básicos como cadastro de produtos e quantidade de estoque, além de consultar os pedidos realizados pelo sistema. O cliente, por sua vez, é quem vai realizar os pedidos, cadastrar novos clientes e consultar preços.

A figura 5 demonstra como as partes interagem entre si, formando o sistema de geração de pedidos e controle de estoque, em um ambiente distribuído geograficamente.

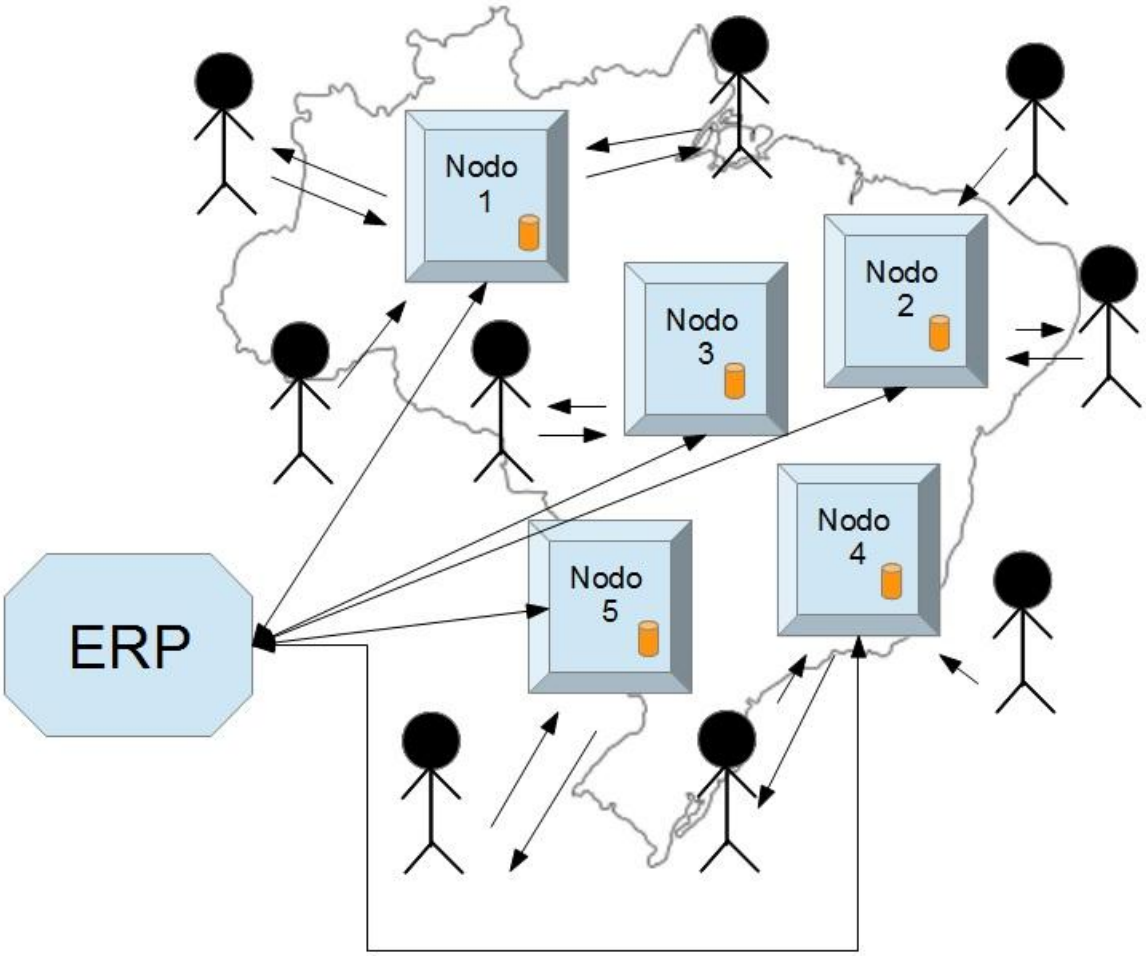

Figura 5 - Esquema Geral do Sistema

4.2. Levantamento de Requisitos

Para auxiliar a esclarecer o escopo do sistema, é feito um levantamento dos requisitos que o sistema deve contemplar.

- O sistema deverá receber informações de entrada e saída de estoque do ERP vinculado a ele.
- O sistema fornecerá os pedidos gerados sempre que solicitado pelo ERP e os apagará de sua base de dados assim que o recebimento destes seja

confirmado, pois o gerenciamento de pedidos é função do ERP, e guardar uma cópia deles no sistema ocuparia espaço desnecessário.

- O pedido gerado conterá informações do cliente, condição de pagamento e produtos.
- O sistema deverá permitir a inserção de um novo servidor apenas configurando-o como nodo integrante, desde que este servidor possua requisitos para rodar o sistema.
- O sistema deve gerar pedidos de forma a garantir que nenhum pedido confirmado contenha um produto que não conste no estoque do sistema. Qualquer nodo está apto a realizar o pedido.

4.3. Casos de Uso

Para o escopo deste projeto, não é necessária uma quantidade muito grande de casos de uso. A expansão do sistema é objeto de estudo de outros projetos. Sendo assim, o foco é o desenvolvimento de funcionalidades que permitam a geração dos pedidos.

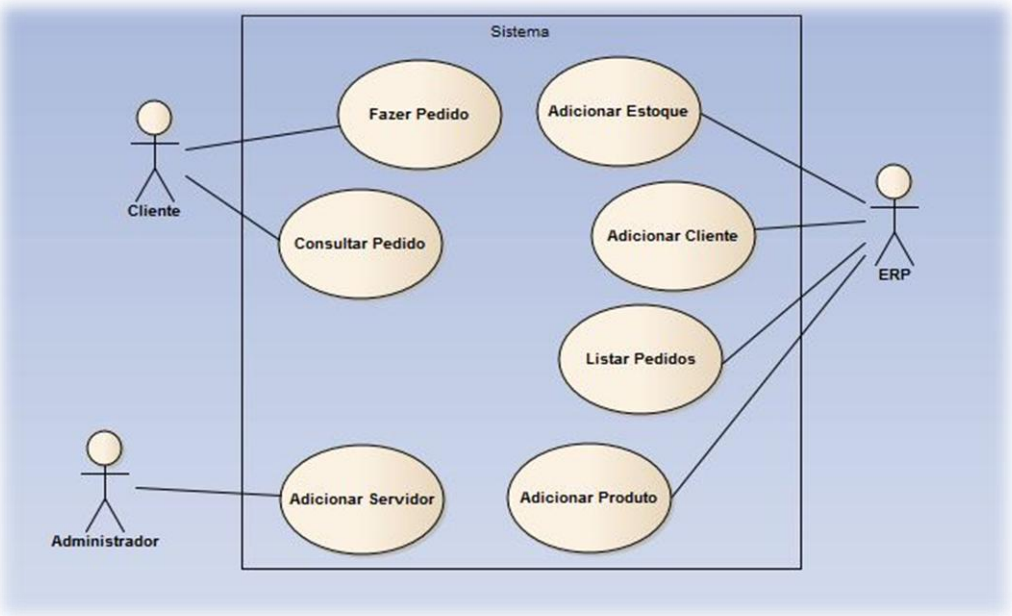

Figura 6 - Casos de Uso

Caso de Uso: Fazer Pedido

Ator: Cliente

Descrição: O cliente seleciona os produtos que deseja comprar, seleciona uma forma de pagamento, seleciona o endereço de entrega padrão para ele ou digita outro endereço de entrega. Após analisar as informações, o cliente confirma o pedido e espera a confirmação do sistema.

Caso de Uso: Consultar Pedido

Ator: Cliente

Descrição: O cliente informa o número do pedido que deseja consultar e o sistema retorna os dados referentes a este pedido.

Caso de Uso: Adicionar Servidor

Ator: Administrador

Descrição: O Administrador do sistema informa o endereço do novo servidor, que já deve estar com o sistema instalado e conectado na rede. O sistema testa este novo servidor. Se o candidato a servidor responder corretamente ele será adicionado. Senão, o sistema retornará uma mensagem de erro.

Caso de Uso: Adicionar Estoque

Ator: ERP

Descrição: O ERP envia para o sistema uma lista contendo códigos de produtos e as respectivas quantidades para incrementar seu estoque.

Caso de Uso: Adicionar Cliente

Ator: ERP

Descrição: O sistema ERP envia para o sistema os dados do novo cliente. Se o CPF ou CNPJ deste cliente já existirem no sistema a operação é abortada. Senão, o sistema inclui o cliente e retorna uma mensagem de sucesso.

Caso de Uso: Listar Pedidos

Ator: ERP

Descrição: O ERP envia uma solicitação para o sistema contendo um intervalo de datas e/ou o CPF ou CNPJ do cliente. O sistema então retorna uma lista contendo todos os pedidos que se encaixam nestes parâmetros. Ao receber os pedidos, o ERP manda uma mensagem de confirmação ao sistema para que este apague os pedidos entregues de sua base de dados.

Caso de Uso: Adicionar Produto

Ator: ERP

Descrição: O ERP envia para o sistema os dados do produto e espera a confirmação de inclusão.

4.4. Classes

Como o sistema implementa o paradigma de Orientação a Objetos, suportado fortemente pela linguagem escolhida, ele possui um conjunto de classes, distribuídas entre os projetos que o formam, classificadas de acordo com suas funcionalidades.

4.4.1. Classes Objeto

As classes pertencentes a este conjunto não possuem métodos nem funções, apenas propriedades. Elas servem como base para a manipulação de objetos tanto na troca de dados do nodo com o mundo externo quanto nas funcionalidades internas do nodo. A figura a seguir mostra o diagrama de classes deste grupo.

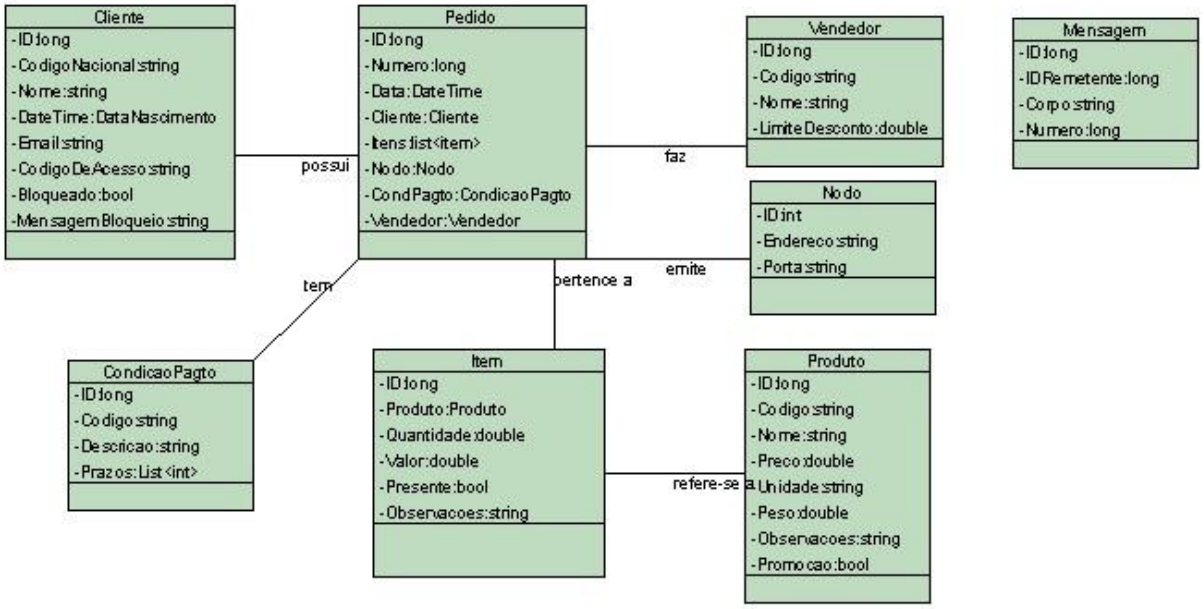

Figura 7- Diagrama das Classes Objeto

4.4.2. Classes de Implementação

Estas classes são extensões das classes objeto, incluindo as funções e métodos necessários para a implementação no projeto Dealer. Basicamente as classes de implementação herdam todas as propriedades das classes objeto, e adicionam as funcionalidades necessárias para aquele objeto. Como padrão o nome destas classes iniciam com o prefixo “Imp” concatenado com o nome da classe objeto referente. Cada classe de implementação possui um conjunto próprio de regras implementadas.

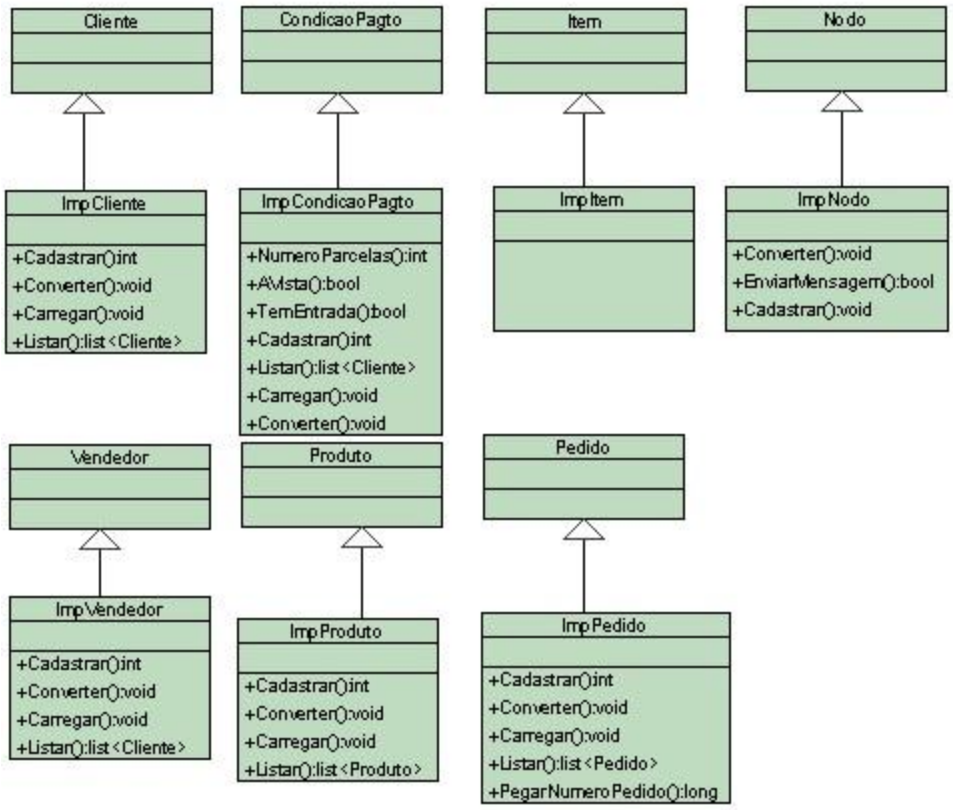

Figura 8 - Diagrama das Classes de Implementação

4.4.3. Classes de Persistência

Este conjunto de classes é gerado pelo DBClassMapper, descrito na seção 3.12, para possibilitar a manipulação de dados no banco de dados através da ferramenta ORM GenDal referenciada na seção 3.11. O nome destas classes é o igual ao nome da tabela correspondente no banco de dados precedido pelo prefixo “DB”. No cabeçalho da classe existe um atributo utilizado para mapear a classe, assim como cada propriedade, mapeada para uma coluna desta tabela.

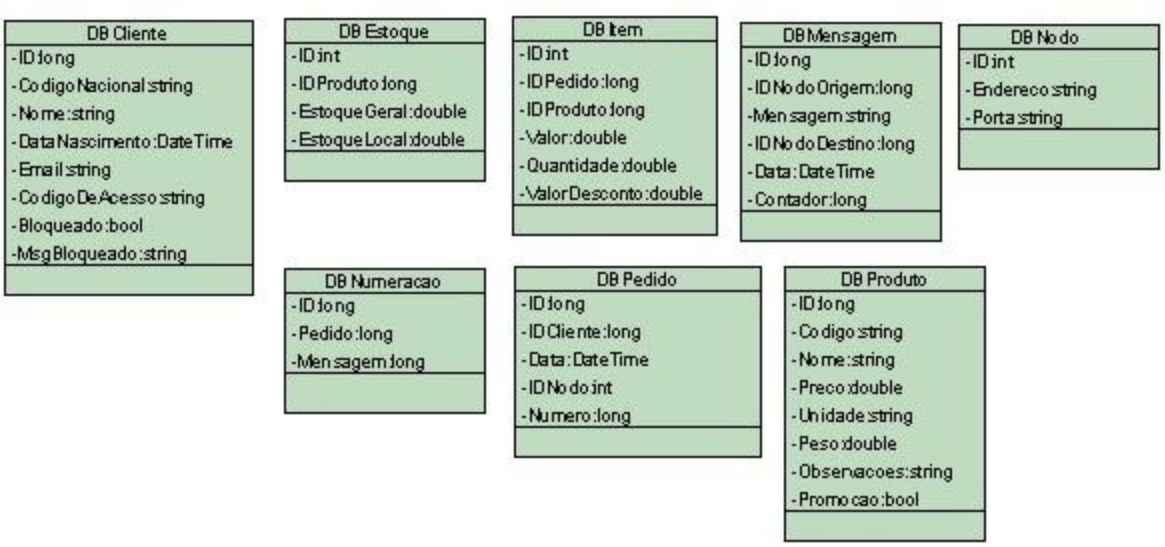

Figura 9 - Diagrama das Classes de Persistência

4.5. Esquema do Banco de Dados

Cada servidor contará com um banco de dados idêntico aos demais. Nestes bancos serão gravados os pedidos gerados, um log de mensagens executadas e uma lista com os outros servidores ativos.

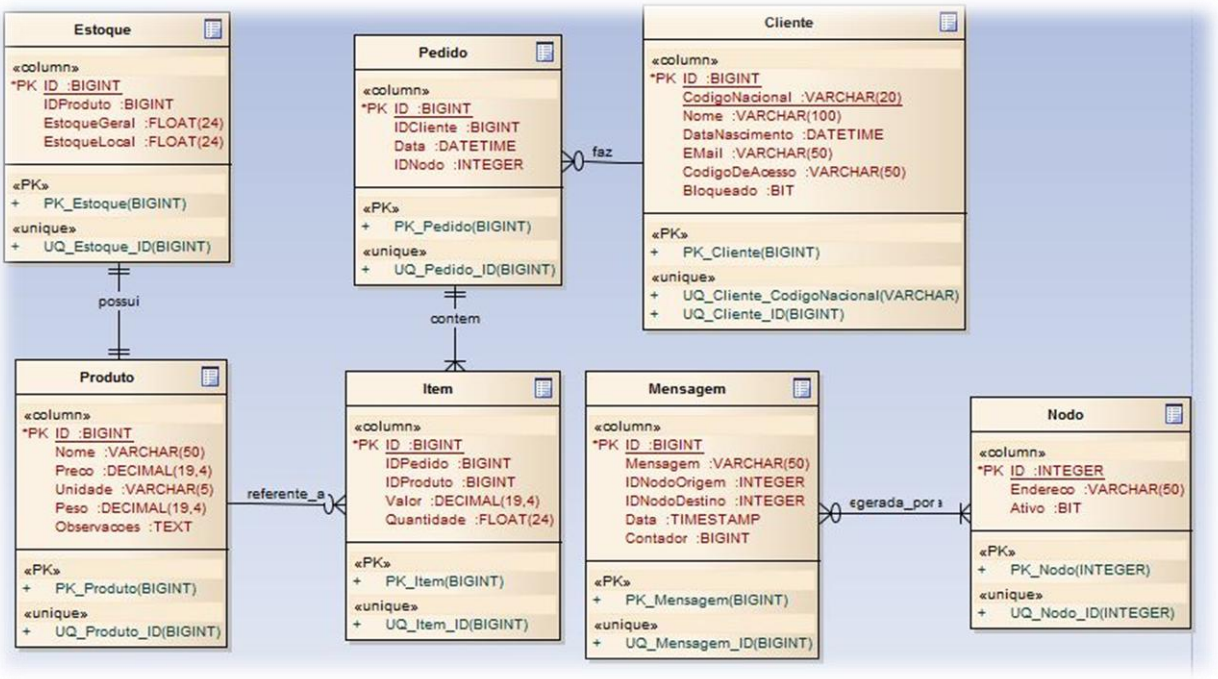

Figura 10 - Diagrama de Tabelas do Banco de Dados

Os tipos de dados podem variar um pouco dependendo do SGBD utilizado em conjunto com o sistema.

4.6. Estrutura do Sistema

O sistema conta com um grupo de servidores espalhados geograficamente e ligados entre si através de uma rede de longa distância. Cada servidor executa exatamente as mesmas funcionalidades, que são:

- Criação de pedidos
- Verificação de nodos vivos
- Incremento de Estoque

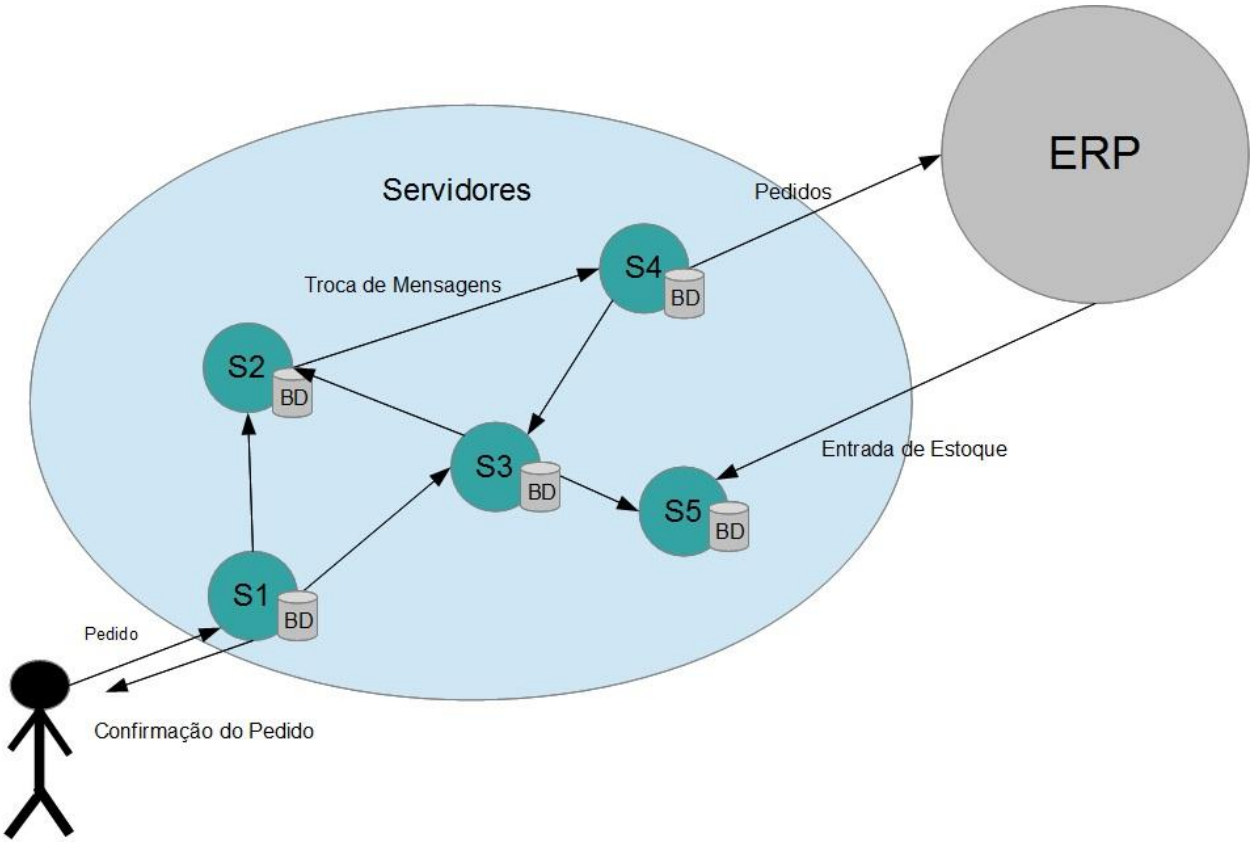

Figura 11 - Diagrama Geral de Funcionamento do Sistema

Cada servidor possui uma base de dados com exatamente o mesmo esquema. O sistema possui um estoque geral de cada produto (conhecido por todos os nodos) e cada nodo (servidor) possui um estoque local. Um pedido só pode ser realizado em um determinado servidor se este possuir estoque local suficiente. Se o nodo não possuir estoque local, mas o estoque global estiver com quantidade suficiente, ele enviará uma mensagem solicitando transferências de um ou mais nodos. Quando um nodo recebe uma solicitação de transferência, ele diminui uma quantidade x de seu estoque e o nodo solicitante aumenta x em seu estoque local. Se o nodo requisitado possui estoque

suficiente, ele transferirá o solicitado e mais uma quantidade estipulada através do cálculo descrito em detalhes na seção 4.9. Caso contrário, ele fará uma transferência de todo o seu estoque local.

Todas as mensagens trocadas serão gravadas no *log* de cada máquina com seus respectivos contadores. Quando um servidor cai é feita uma transferência automática de seu estoque local para um servidor vivo. Ao retornar à atividade, um servidor deve processar todas as mensagens que foram perdidas durante sua ausência, inclusive as transferências automáticas.

4.7. Projetos de Implementação

A implementação deste sistema depende de 3 projetos principais: Barsa, Dealer e Sherlock e conta com a criação de 2 projetos que desempenham o papel de cliente do sistema: cliente Windows e o cliente Android.

4.7.1. Projeto Barsa

O projeto Barsa foi implementado no framework .NET utilizando a linguagem C# [10]. Ele é compilado em forma de DLL (*Dynamic-Link Library*) e é referenciado em todos os outros projetos. O projeto Barsa é um conjunto de todas as classes primitivas utilizadas no sistema. As classes definidas no projeto Barsa possuem apenas propriedades, não têm nenhum método definido, pois servem apenas para a criação de objetos básicos, que servem para a troca de mensagens, chamadas de web services e nos métodos internos do sistema.

4.7.2. Projeto Dealer

O projeto Dealer possui um web service responsável por toda a comunicação do nodo com o mundo exterior. Além disso, o Dealer é responsável pela persistência de dados, através do uso do GenDal, descrito na seção 3.11. Ele possui um conjunto de classes de persistência, um conjunto de classes de implementação e a classe Correios.

As classes de persistência são criadas com o auxílio da interface do DBClassMapper, para as manipulações de banco de dados. Como padrão, o nome das classes é a concatenação da sigla “DB” com o nome da tabela do banco de dados vinculada a esta classe. Por exemplo, a classe DBCliente é um mapeamento GenDal para a tabela Cliente do banco de dados. Todas as classes de persistência importam os *namespaces* DBManager.DBMapper, System.ComponentModel.DataAnnotations e System. No cabeçalho, elas possuem um indicador da Tabela, e cada propriedade da classe representa uma coluna da tabela no banco de dados. A principal vantagem de trabalhar com um ORM, como é o caso do GenDal, é a portabilidade, em relação a banco de dados que o sistema possui. No caso deste projeto, o banco de dados utilizado é o Microsoft SQL Server 2008, mas a escolha do SGBD é irrelevante dentro do projeto, desde que ele seja suportado pelo GenDal. Para exemplificar, o quadro 2 contém o código da classe DBCliente:

```
using DBManager.DBMapper;
using System.ComponentModel.DataAnnotations;
using System;
namespace Dealer
{
    [DBTable("dbo", "Cliente")]
    public class DBCliente
    {
    #region "DBCliente_Fields"
        private long? iD;
        private string codigoNacional;
        [DBField()]
        [StringLength(50, ErrorMessage="Property Email exceeded it's limits(0-50).")]
        public string Email
        {
            get { return email; }
            set { email = value; }
        }
        [DBField()]
        [StringLength(50, ErrorMessage="Property CodigoDeAcesso exceeded it's limits(0-50).")]
```

```
        private string nome;
        private DateTime dataNascimento;
        private string email;
        private string codigoDeAcesso;
        private bool bloqueado;
        private string msgBloqueado;
    #endregion

    #region "DBCliente_Properties"
        [DBField()]
        [Key()]
        [Editable(false)]
        public long? ID
        {
            get { return iD; }
            set { iD = value; }
        }

        [DBField()]
        [DBAlternateKey]
        [StringLength(20, ErrorMessage="Property
CodigoNacional exceeded it's limits(0-20).")]
        public string CodigoNacional
        {
            get { return codigoNacional; }
            set { codigoNacional = value; }
        }
        [DBField()]
        [StringLength(100, ErrorMessage="Property Nome
exceeded it's limits(0-100).")]
        public string Nome
        {
            get { return nome; }
            set { nome = value; }
        }
        [DBField()]
        [DBDateTime()]
        public DateTime DataNascimento
        {
            get { return dataNascimento; }
            set { dataNascimento = value; }
        }
        public string CodigoDeAcesso
        {
            get { return codigoDeAcesso; }
            set { codigoDeAcesso = value; }
        }
        [DBField()]
        public bool Bloqueado
        {
            get { return bloqueado; }
            set { bloqueado = value; }
        }

        [DBField()]
        public string MsgBloqueado
        {
            get { return msgBloqueado; }
            set { msgBloqueado = value; }
        }
    #endregion
    }
}
```

Quadro 2 - Exemplo de Classe de Interação com Gendal

As classes de implementação contêm toda a lógica relacionada a cada objeto básico contido no projeto Barsa. Cada classe de implementação herda de uma classe básica, acrescentando as funcionalidades necessárias para o desempenho de cada objeto dentro do sistema. Os métodos mais comuns são: Cadastrar(), Listar(), Carregar(). O método Cadastrar é responsável por incluir e/ou editar no banco de dados um registro

referente àquele objeto; Listar retorna uma lista de objetos básicos do banco de dados utilizando as propriedades do objeto para filtrar os dados; Carregar preenche o próprio objeto com os dados obtidos no banco de dados, utilizando como parâmetro o identificador do objeto. Além disso, algumas classes de implementação têm alguns métodos específicos, como é o caso da classe de implementação do pedido, que possui o método PegarNumeroPedido, responsável por buscar um número único para um novo pedido.

A classe Correios é a única classe "*static*" do projeto. Ela reúne as informações necessárias para a integração dos nodos, formando a rede que provê as funcionalidades requeridas na especificação do sistema. A classe Correios possui uma lista de todos os nodos conhecidos, as informações do próprio nodo e a constante string de conexão (feita através de ODBC, que busca como padrão as configurações nomeadas como DealerDataBase). Seu principal método é o Distribuir, que recebe uma mensagem, inclui os dados necessários de comunicação e manda para cada nodo conhecido, inclusive para ele mesmo.

4.7.3. Projeto Sherlock

Este projeto é a implementação de um serviço que roda em *background* em cada nodo responsável por descobrir a existência de novos nodos e a queda de nodos existentes. O projeto Sherlock tem a função apenas de atualizar a listagem de nodos do Dealer, e este sim é quem toma as decisões do que fazer. Para isso, o Sherlock foi construído como um Windows Service, configurado em cada nodo para rodar em *background* inicializando-se no momento em que a máquina inicia o seu funcionamento. O

funcionamento do Sherlock é muito simples: um serviço que fica de tempos em tempos enviando uma solicitação SOAP para todos os nodos conhecidos, perguntando para eles quais os nodos que eles conhecem. Desta forma o Sherlock investiga novos nodos do sistema e ao mesmo tempo detecta falhas em outros nodos.

4.7.4. Cliente Windows

Os projetos clientes foram feitos apenas com o intuito de demonstrar o funcionamento do sistema. Na prática o Cliente Windows faz o papel de ERP comunicando-se com o sistema e também papel de cliente (com a criação de pedidos). O cliente Windows é uma aplicação feita utilizando o template Windows Forms, que possui uma interface que permite o cadastro de nodos, o cadastro de clientes, produtos e pedidos; e a consulta dos itens cadastrados.

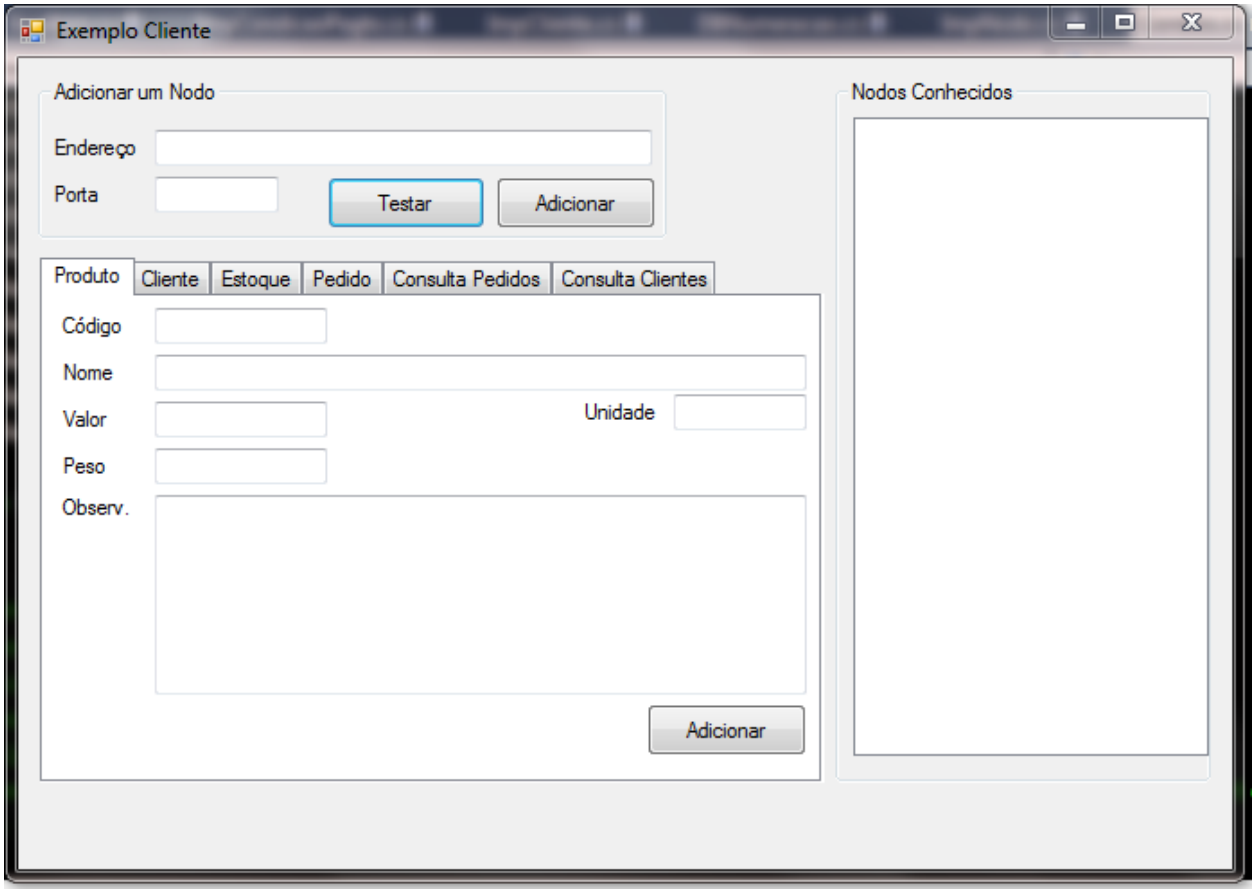

Figura 12 - Interface do Cliente Windows

O funcionamento desta aplicação é muito simples: existe uma referência web, que dependendo dos dados fornecidos pelo usuário, aponta para um determinado nodo do sistema. Cada tela possui um formulário e um botão que, quando acionado, envia a solicitação ao nodo, que a processa e entrega o resultado para o cliente. Na prática, o cliente só precisa de uma comunicação operante com o web service do nodo e uma interface com o usuário final.

4.7.5. Cliente Android

O Cliente Android é um projeto com as funcionalidades de cliente, desenvolvido para rodar na plataforma Android. Este projeto foi implementado utilizando a IDE Eclipse e o *framework* PhoneGap, descrito na seção 3.6. A geração de pedidos é feita através de uma interface onde o usuário seleciona os produtos através de uma listagem e manda este pedido para o sistema gerenciador de dados, indicando os dados de cliente, produtos e quantidades. A interface avisará se a resposta do sistema indica que o pedido foi gravado com sucesso ou se houve algum tipo de falha (técnica ou de falta de estoque).

4.8. Funcionamento do Nodo

Cada nodo necessita possuir um servidor IIS funcional, onde o Dealer estará instalado e com o web service acessível em uma porta para os outros nodos. Cada nodo é idêntico, em relação ao sistema, aos demais, ou seja, não existe distinção ente eles. O nodo, individualmente, possui acesso à DLL do GenDal e à DLL do projeto Barsa. Além disso, possui um banco de dados próprio, cujo esquema é igual aos demais nodos, e um serviço Sherlock funcional.

Normalmente, um nodo corresponde a uma máquina física, porém nada impede que em um mesmo computador sejam instalados vários nodos, desde que cada um tenha uma porta de acesso única para o web service, um banco de dados individual e um Sherlock configurado para o seu serviço.

O nodo se comunica com o ambiente externo e também com outros nodos. A comunicação com o ambiente externo se dá via chamadas do web service, respeitando as funções disponibilizadas pelo nodo. Já a comunicação entre os nodos é feita via

troca de mensagens implementando o controle de precedência através do algoritmo de Lamport, conforme descrito por Coulouris [1]. O algoritmo do Lamport funciona da seguinte maneira: cada processo do sistema distribuído mantém um contador crescente e monotônico. Cada evento do processo recebe este contador, previamente incrementado. Quando um processo recebe um evento cujo contador seja maior que o seu, seu contador terá o valor deste processo incrementado. Desta forma os contadores mantém uma relação temporal dos eventos.

Além disso, os nodos contam com uma tabela para a gravação das mensagens recebidas e enviadas, formando o log de mensagens mencionado anteriormente. Quando um servidor cai, para retomar um estado válido dentro do sistema, o log é verificado, assim o nodo sabe quais mensagens foram executadas e ele pode voltar a funcionar normalmente assim que processar as outras mensagens.

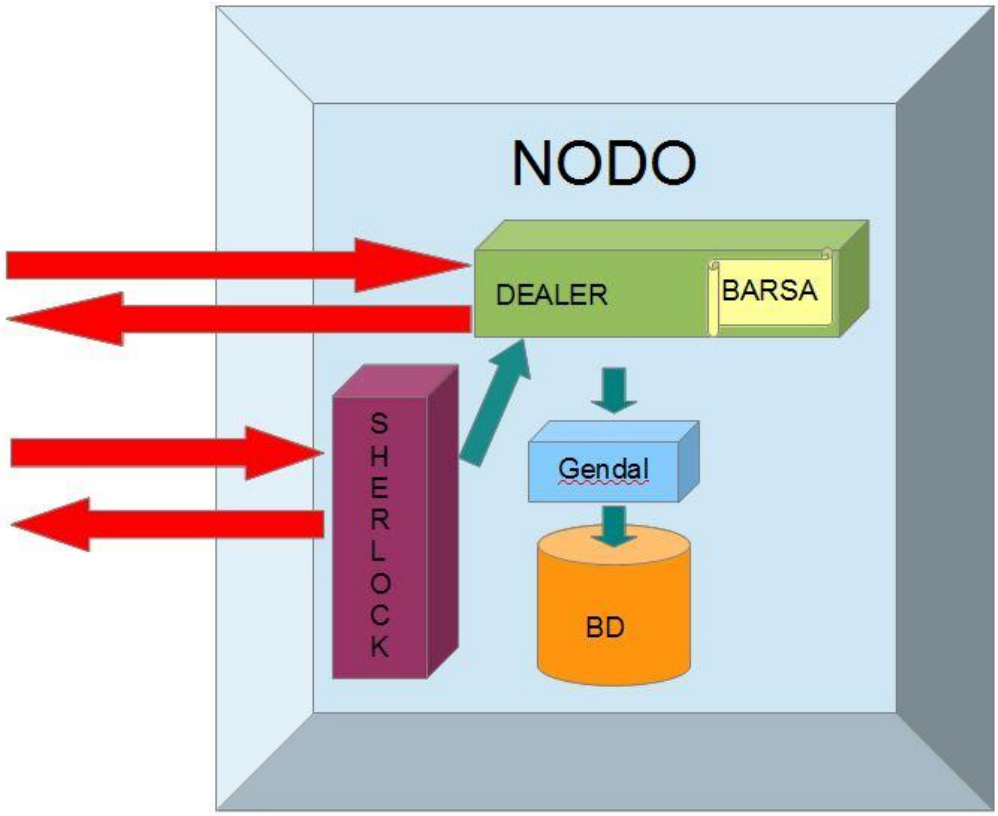

Figura 13 - Esquema do Nodo – As setas vermelhas indicam o fluxo de dados do nodo entre o nodo e o mundo exterior. A setas azuis representam as trocas de mensagens internas do nodo.

4.9. Sistemática do Controle de Estoque

Como já explicado, a função deste sistema é a criação de pedidos de maneira distribuída. Dois itens devem ser necessariamente contemplados para que o sistema possa ter sua utilização justificada: a velocidade das consultas e gravações de um pedido em ambiente com restrições de comunicação deve ser maior em relação a um modelo cliente-servidor simples; e o sistema não pode, de nenhuma maneira, vender um item que não consta no estoque.

Para que haja um controle de estoque, em um sistema que roda paralelamente em vários nodos, foi adotada uma estratégia que visa resolver este problema sem causar um grande *overhead* no sistema, demonstrada na Figura 14.

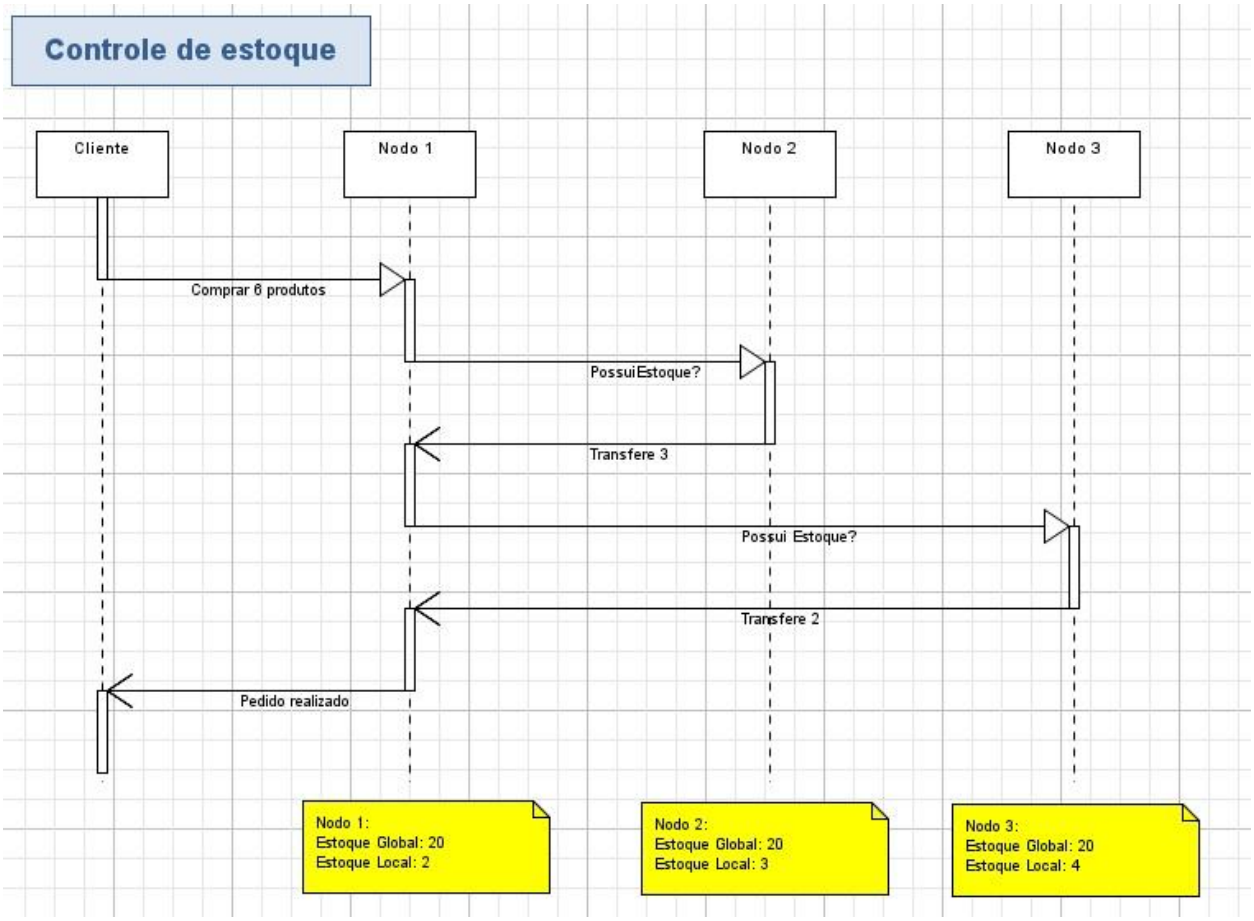

Figura 14 - Diagrama de sequência do controle de estoque no momento do pedido

Existem neste sistema dois estoques de cada produto: O estoque geral e o estoque local. O estoque geral é a quantidade total disponível deste produto no sistema, e representa a quantidade física do produto em estoque. O estoque geral tende a ser o mesmo para um determinado produto em todos os nodos. Como manter esta informação atualizada e confiável em todos os nodos custa muito, a solução

escolhida foi a criação de um estoque local. O estoque local é um dado virtual, não possui nenhuma correlação com o estoque físico do produto e não é de conhecimento do cliente. Cada nodo possui além do estoque geral, o estoque local, que é a quantidade disponível de venda de um determinado produto para aquele nodo. Se uma solicitação de pedido possui uma quantidade menor ou igual à quantidade disponível no estoque local do nodo que está atendendo esta requisição, o pedido será feito, sem a necessidade prévia de nenhum outro nodo. Os demais nodos somente serão avisados sobre o pedido após a efetivação deste, sem atrasos para o cliente. Se a solicitação de pedido possuir uma quantidade maior que o estoque local e menor ou igual que o estoque global, o nodo vai perguntar a todos os seus conhecidos se eles possuem estoque local daquele produto, por ordem de menor ping. Quando encontrar este nodo, é realizada uma transferência de estoque local. A quantidade transferida é determinada pelo seguinte cálculo, tomando como variáveis p: quantidade solicitada, n1: quantidade do nodo 1, n2: quantidade do nodo2, sendo nodo 1 o requisitante e nodo 2 como requisitado:

Se (p – n1) >= n2, a quantidade transferida será n2

Se (p – n1) < n2, a quantidade transferida será (p-n1) + k, sendo k = (n2 - (p-n1)) / 2

Caso k não seja um número inteiro, k será o maior inteiro menor que k.

Esta transferência de estoque maior que o necessário para o pedido é feita para que haja uma distribuição do estoque entre os nodos que mais vendem um

determinado produto e diminuição gradual dos pedidos que necessitam de transferência para serem concluídos.

Quando um nodo é declarado falho, os outros nodos fazem uma transferência automática de todo o estoque local presumido daquele nodo. Esta transferência é feita quando um nodo se declara responsável e recebe uma resposta positiva de 50% mais um dos outros nodos. O estoque presumido é feito bloqueando o sistema para qualquer alteração de estoque daquele produto, somando todos os estoques locais e tirando a diferença do estoque global. O sistema é desbloqueado assim que a transferência é feita. Se o nodo eleito morrer durante o processo, as transferências serão refeitas repetindo o processo. Se este nodo voltar à ativa, na recuperação de mensagens do *log* constarão todas as transferências e consequentemente este nodo iniciará sem estoque local. No caso de um nodo novo, todos os produtos também estarão com estoques locais zerados.

4.10. Gravação do Pedido

Quando um nodo recebe a solicitação para a gravação de um pedido, após a verificação das quantidades de cada produto através do conjunto de regras escritos na seção 4.9, o próximo passo é notificar os nodos conhecidos sobre o pedido. O pedido só vai ser de fato consolidado quando o nodo responsável por dele receber pelo menos uma mensagem de conhecimento (*ack*) de um outro nodo para o qual a geração deste pedido foi comunicada. Assim que tiver esta confirmação, o pedido é gravado e a aplicação cliente recebe uma confirmação, através do web service de que este pedido

foi gravado com sucesso. Caso haja algum problema dentro deste processo, a aplicação cliente receberá um código de erro.

Este controle é necessário para que, caso haja uma falha de algum nodo, seus pedidos não fiquem perdidos. Quando o ERP acessar o sistema, ele deve ter acesso a todos os pedidos feitos, inclusive se o nodo em questão estiver em falha. Quando o nodo manda um pedido para o ERP ele exclui este pedido de seu banco de dados, e manda esta mensagem para os outros nodos.

5. Estudo de Caso

Para exemplificar o uso do sistema será apresentado um estudo de caso baseado na empresa fictícia “1000 Sapatos”. Esta empresa é uma grande fabricante de sapatos, que possui representantes em todo o território nacional. Para melhorar o acesso dos representantes, a “1000 Sapatos” instalou um servidor em cada região do Brasil, conforme a figura.

Figura 15 - Distribuição dos Nodos

Neste caso, os usuários dos sistemas cliente serão os representantes, que terão a opção de um sistema cliente para Windows, um sistema para aparelhos na plataforma Android e um sistema para aparelhos na plataforma iOS (não implementado neste

projeto). As cidades escolhidas para abrigar os servidores, que desempenham o papel de nodo do sistema são: Florianópolis, São Paulo, Brasília, Salvador e Manaus.

Cada dispositivo cliente operado por um representante possui uma lista de nodos preferenciais, escolhidos de forma a optar sempre pelo menor tempo de comunicação entre os nodos. O representante fictício João atua nas cidades: Porto Alegre/RS, Canoas/RS, Caxias do Sul/RS, Araranguá/SC e Criciúma/SC. O sistema cliente que ele usa deve estar configurado para acessar preferencialmente o nodo Florianópolis, pois é o mais próximo de sua área de atuação. Porém se o sistema cliente não conseguir se comunicar com o nodo Florianópolis ele tentará comunicação com os nodos São Paulo, Brasília, Salvador e Manaus, nesta ordem.

O ERP responsável pelo gerenciamento da empresa encontra-se em um servidor em Belo Horizonte/MG, onde se situa a sede da empresa. O ERP comunica-se com o sistema de pedidos preferencialmente através do nodo São Paulo, e em caso de falha deste nodo tenta se comunicar com os nodos Brasília, Salvador, Florianópolis e Manaus, nesta ordem de preferência. Todo dia, o ERP manda para o sistema de pedidos os novos produtos cadastrados, os produtos com cadastro alterado (incluindo alterações de preço), os clientes novos e os alterados, os vendedores (conhecidos neste caso como representantes) novos e alterados. Além disso, o ERP pega os pedidos cadastrados e os grava em sua própria base de dados para futura emissão de nota fiscal e envio da mercadoria.

O representante João trabalha com o cliente Android, e deseja realizar um pedido contendo os seguintes produtos com as respectivas quantidades:

Código	Nome do Produto	Qtd Desejada	Est. Geral	Est.Fpolis	Est.S.Paulo	Est.Brasília
152	Bota Fem. #154	15	35	10	9	8
101	Sapato Masc. #5	2	100	17	58	5
859	Sandalia #1547	18	17	1	4	2
785	Tênis Infantil #9	5	30	2	2	10

Quadro 3 - Quantidades e Estoques

Quando o sistema recebe esta solicitação de pedido, supondo que o cliente Android não faça nenhuma restrição de estoque, ele analisa separadamente cada produto solicitado para poder validar o estoque.

O produto de código 152 possui um estoque geral de 35 unidades. Como a 1000 Sapatos possui apenas um estoque físico de produtos supõe-se que existam 35 unidades deste produto disponíveis para venda em todo o país. Porém o nodo Florianópolis possui um estoque local (quantidade virtual para venda) de apenas 10 unidades, não satisfazendo totalmente a solicitação deste produto. Como teoricamente o sistema possui uma quantidade suficiente deste produto, o nodo Florianópolis solicita uma transferência de estoque local para São Paulo. Lembrando que esta transferência é lógica apenas, ou seja, nenhum produto físico estará sendo transportado. O nodo Florianópolis solicita 5 unidades do produto 152 para o nodo São Paulo. Aplicando a regra de transferência de estoque citada no capítulo 4.9, como o nodo São Paulo possui 9 unidades no estoque local ele fará uma transferência de 7 unidades para Florianópolis. Desta forma São Paulo ficará com um estoque local de 2 unidades e Florianópolis, 17. Após a transferência, o produto 152 poderá ser vendido conforme solicitado.

O produto 101 não terá que solicitar quaisquer informações dos outros nodos, já que o estoque local do nodo Florianópolis possui quantidade suficiente para o pedido em questão.

Como a quantidade solicitada do produto 859 é maior que o estoque geral (ou seja, não existe quantidade física deste produto disponível para a venda), este pedido não poderá ser realizado com esta quantidade. Um código de erro será enviado para a aplicação cliente.

Analisando o caso do produto 785, verificamos que o sistema terá que solicitar transferência tanto de São Paulo quanto de Brasília para que o pedido possa ser realizado.

6. Conclusão

Trabalhar com um sistema distribuído resulta em uma complexidade bem maior de software se comparado ao modelo clássico cliente-servidor. O custo porém é justificado em casos específicos, como é o caso analisado neste trabalho. A vantagem de um sistema distribuído neste contexto se dá quando há uma dificuldade em unificar um sistema de pedidos, seja por falta de infraestrutura de rede ou a formação de um gargalo em torno de um servidor centralizado.

A solução implementada neste trabalho mostra como é possível dividir as funcionalidades, de forma a atender melhor as partes mais críticas de um sistema de gerenciamento através de um sistema especialista e facilmente integrável com outros sistemas. Apesar de ser uma solução direcionada a um caso bem específico, pode servir como base para vários problemas comuns encontrado em empresas grandes, principalmente em países onde a infraestrutura tecnológica ainda representa um entrave no desenvolvimento empresarial.

No decorrer deste projeto percebe-se o quão importante é utilizar padrões bem conhecidos em diversas tecnologias, para limitar o mínimo possível os requisitos tecnológicos exigidos no funcionamento do sistema. A forma como foi elaborado, faz com que este sistema seja uma caixa preta com uma comunicação externa padronizada (Web Services), o que possibilita uma integração facilitada com grande parte das plataformas existentes, facilitando a absorção deste sistema pelo mercado.

6.1. Trabalhos Futuros

Apesar do sistema implementado neste trabalho ser aplicável em casos reais, alguns itens poderão ser alvo de trabalhos futuros a fim de ampliar as funcionalidades, e consequentemente a utilização do mesmo no mercado real.

Um dos pontos que podem ser alvo de pesquisa é a questão da segurança das informações que trafegam dentro do sistema. Algumas empresas teriam este ponto como pré-requisito para a utilização, já que teriam que proteger seus dados por questões estratégicas.

Algumas funcionalidades interessantes não foram abordadas neste trabalho, mas podem ser exploradas futuramente, como o controle de múltiplos almoxarifados e de representantes de vendas.

7. Bibliografia

1. COULOURIS, George; DOLLIMORE, Jean; KINDBERG, Tim. **Sistemas Distribuídos Conceitos e Projetos**, quarta edição, Editora Bookman.

2. The official IIS Site – http://www.iis.net. Acesso em 16/10/2012

3. BECKER, CLARO e SOBRAL. **Web Services e XML , Um Novo Paradigma da Computação Distribuída**.Disponível em http://homes.dcc.ufba.br/~dclaro/download/ArtigoWebServices.pdf

4. BERGAMASCHI, Sidnei. **Um estudo sobre projetos de implementação de sistemas para gestão empresarial**. 1999. Dissertação (Mestrado em Métodos Quantitativos) - Faculdade de Economia, Administração e Contabilidade, Universidade de São Paulo, São Paulo, 1999. Disponível em: <http://www.teses.usp.br/teses/disponiveis/12/12133/tde-27122003-224740/>. Acesso em: 2012-07-03.

5. Da SILVA, Diego Magno. **Ferramenta de Manipulação da Biblioteca GenDAL para mapeamento Objeto-Relacional visando a melhoria na interação com o usuário.** 2012. Projeto de conclusão do curso de Ciências da Computação UFSC, Florianópolis, Santa Catarina. (no Prelo)

6. PhoneGap Site official do framework – http://phonegap.com/. Acesso em 01/11/2012

7. JQuery. Site oficial do framework. Disponível em: http://jquerymobile.com/. Acesso em 01/11/2012.

8. SPIER, Fernanda. **Comparativos de frameworks para desenvolvimento mobile multiplataforma**. 2012. Anteprojeto de trabalho de conclusão do curso de Sistemas da Informação da Universidade Feevale, Novo Hamburgo, Rio Grande do Sul.

9. FOWLER, Martin. **UML Essencial.** Terceira edição. Editora Bookman

10. ROBINSON, Simon; NAGEL, Christian; WATSON, Karli; GLYNN, Jay; SKINNER, Morgan; EVJEN, Bill. **Professional C#.** Terceira edição, editora Wrox.

11. C#.Site Oficial.Disponível em: http://msdn.microsoft.com/library/vstudio/z1zx9t92. Acesso em 06/11/2012.

12. SQL Server. Site Oficial. Disponível em: http://www.microsoft.com/sqlserver/en/us/default.aspx. Acesso em 17/11/2012

13. GOODMAN, Danny. **Java Script a Bíblia**. Primeira Edição. Editora Campus.

14. LEANDRO, Sabrina da Silva. **Balanceamento de Carga em Web Services.** 2005. Trabalho de conclusão do curso de Sistemas de Informação. Universidade Federal de Santa Catarina.

APÊNDICE A – Código fonte da aplicação desenvolvida

Barsa

Cliente.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Barsa
{
    public class Cliente
    {
        public long? ID { get; set; }
        public string CodigoNacional { get; set; } //CPF ou CNPJ
        public string Nome { get; set; }
        public DateTime DataNascimento { get; set; }
        public string EMail { get; set; }
        public string CodigoDeAcesso { get; set; }
        public bool Bloqueado { get; set; }
        public string MensagemBloqueio { get; set; }
    }
}
```

CondicaoPagto.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Barsa
{
    public class CondicaoPagto
    {
        public long ID { get; set; }
        public string Codigo { get; set; }
        public string Descricao { get; set; }
        public List<int> Prazos { get; set; }

    }
}
```

Funcoes.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Xml;
using System.Xml.Serialization;
using System.IO;

namespace Barsa
{
   public static class Funcoes
   {
      public static String Serializar(object o)
      {
         if (o != null)
         {

            XmlSerializer serializer = new XmlSerializer(o.GetType());

            MemoryStream ms = new MemoryStream();

            serializer.Serialize(ms, o);

            return Encoding.UTF8.GetString(ms.GetBuffer());

         }

         return null;
      }

      public static T Deserializar<T>(String xml)
      {
         T returnedXmlClass = default(T);

         try
         {
            using (TextReader reader = new StringReader(xml))
            {
               try
               {
                  returnedXmlClass = (T)new XmlSerializer(typeof(T)).Deserialize(reader);
               }
               catch (InvalidOperationException)
               {
                  // String passed is not XML, simply return defaultXmlClass
               }
            }
         }
         catch (Exception ex)
         {
         }
```

```
            return returnedXmlClass;
        }
    }
}
```

Item.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Barsa
{
    public class Item
    {
        public long? ID { get; set; }
        public Produto Produto { get; set; }
        public double Quantidade { get; set; }
        public double Valor { get; set; }
        public bool Presente { get; set; }
        public string Observacoes { get; set; }
    }
}
```

Mensagem.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Barsa
{
    public class Mensagem
    {
        public long ID { get; set; }
        public long IDRemetente { get; set; }
        public string Corpo { get; set; }
        public long Numero { get; set; }
    }
}
```

Nodo.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Barsa
```

```
{
    public class Nodo
    {
        public int ID { get; set; }
        public string Endereco { get; set; }
        public string Porta { get; set; }

    }
}
```

Pedido.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Barsa
{
    public class Pedido
    {
        public long? ID { get; set; }
        public long Numero { get; set; }
        public DateTime Data { get; set; }
        public Cliente Cliente { get; set; }
        public List<Item> Itens { get; set; }
        public Nodo Nodo { get; set; }
        public CondicaoPagto CondPagto { get; set; }
        public Vendedor Vendedor { get; set; }
    }
}
```

Produto.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Barsa
{
    public class Produto
    {
        public long? ID { get; set; }
        public string Codigo { get; set; }
        public string Nome { get; set; }
        public double Preco { get; set; }
        public string Unidade { get; set; }
        public double Peso { get; set; }
        public string Observacoes { get; set; }
        public bool Promocao { get; set; }
    }
}
```

Vendedor.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace Barsa
{
    public class Vendedor
    {
        public long ID { get; set; }
        public string Codigo { get; set; }
        public string Nome { get; set; }
        public double LimiteDesconto { get; set; }
    }
}
```

Dealer

DBCliente.cs

```
using DBManager.DBMapper;
using System.ComponentModel.DataAnnotations;
using System;

namespace Dealer
{
    [DBTable("dbo", "Cliente")]
    public class DBCliente
    {
    #region "DBCliente_Fields"
        private long? iD;
        private string codigoNacional;
        private string nome;
        private DateTime dataNascimento;
        private string email;
        private string codigoDeAcesso;
        private bool bloqueado;
        private string msgBloqueado;

    #endregion

    #region "DBCliente_Properties"

        [DBField()]
        [Key()]
        [Editable(false)]
        public long? ID
        {
            get { return iD; }
```

```
    set { iD = value; }
}

[DBField()]
[DBAlternateKey]
[StringLength(20, ErrorMessage="Property CodigoNacional exceeded it's limits(0-20).")]
public string CodigoNacional
{
    get { return codigoNacional; }
    set { codigoNacional = value; }
}

[DBField()]
[StringLength(100, ErrorMessage="Property Nome exceeded it's limits(0-100).")]
public string Nome
{
    get { return nome; }
    set { nome = value; }
}

[DBField()]
[DBDateTime()]
public DateTime DataNascimento
{
    get { return dataNascimento; }
    set { dataNascimento = value; }
}

[DBField()]
[StringLength(50, ErrorMessage="Property Email exceeded it's limits(0-50).")]
public string Email
{
    get { return email; }
    set { email = value; }
}

[DBField()]
[StringLength(50, ErrorMessage="Property CodigoDeAcesso exceeded it's limits(0-50).")]
public string CodigoDeAcesso
{
    get { return codigoDeAcesso; }
    set { codigoDeAcesso = value; }
}

[DBField()]
public bool Bloqueado
{
    get { return bloqueado; }
    set { bloqueado = value; }
}

[DBField()]
public string MsgBloqueado
{
    get { return msgBloqueado; }
    set { msgBloqueado = value; }
```

```
        }

    #endregion


    }
}
```

DBEstoque.cs

```
using DBManager.DBMapper;
using System.ComponentModel.DataAnnotations;
using System;

namespace Dealer
{
    [DBTable("dbo", "Estoque")]
    public class DBEstoque
    {
    #region "DBEstoque_Fields"
        private int iD;
        private long iDProduto;
        private double estoqueGeral;
        private double estoqueLocal;

    #endregion

    #region "DBEstoque_Properties"

        [DBField()]
        [Key()]
        [Editable(false)]
        public int ID
        {
            get { return iD; }
            set { iD = value; }
        }

        [DBField()]
        public long IDProduto
        {
            get { return iDProduto; }
            set { iDProduto = value; }
        }

        [DBField()]
        public double EstoqueGeral
        {
            get { return estoqueGeral; }
            set { estoqueGeral = value; }
        }

        [DBField()]
        public double EstoqueLocal
```

```
        {
            get { return estoqueLocal; }
            set { estoqueLocal = value; }
        }

    #endregion


    }
}
```

DBItem

```
using DBManager.DBMapper;
using System.ComponentModel.DataAnnotations;
using System;

namespace Dealer
{
    [DBTable("dbo", "Item")]
    public class DBItem
    {
    #region "DBItem_Fields"
        private int?  iD;
        private long iDPedido;
        private long? iDProduto;
        private double valor;
        private double quantidade;
        private double valorDesconto;

    #endregion

    #region "DBItem_Properties"

        [DBField()]
        [Key()]
        [Editable(false)]
        public int? ID
        {
            get { return iD; }
            set { iD = value; }
        }

        [DBField()]
        public long IDPedido
        {
            get { return iDPedido; }
            set { iDPedido = value; }
        }

        [DBField()]
        public long? IDProduto
```

```
        {
            get { return iDProduto; }
            set { iDProduto = value; }
        }

        [DBField()]
        public double Valor
        {
            get { return valor; }
            set { valor = value; }
        }

        [DBField()]
        public double Quantidade
        {
            get { return quantidade; }
            set { quantidade = value; }
        }

        [DBField()]
        public double ValorDesconto
        {
            get { return valorDesconto; }
            set { valorDesconto = value; }
        }

    #endregion


    }
}
```

DBMensagem.cs

```
using DBManager.DBMapper;
using System.ComponentModel.DataAnnotations;
using System;

namespace Dealer
{
    [DBTable("dbo", "DBMensagem")]
    public class DBMensagem
    {
    #region "DBMensagem_Fields"
        private int iD;
        private string mensagem;
        private int iDNodoOrigem;
        private int iDNodoDestino;
        private DateTime data;
        private long contador;

    #endregion

    #region "DBMensagem_Properties"
```

```
        [DBField()]
        [Key()]
        [Editable(false)]
        public int ID
        {
            get { return iD; }
            set { iD = value; }
        }

        [DBField()]
        [StringLength(150, ErrorMessage="Property Mensagem exceeded it's limits(0-150).")]
        public string Mensagem
        {
            get { return mensagem; }
            set { mensagem = value; }
        }

        [DBField()]
        public int IDNodoOrigem
        {
            get { return iDNodoOrigem; }
            set { iDNodoOrigem = value; }
        }

        [DBField()]
        public int IDNodoDestino
        {
            get { return iDNodoDestino; }
            set { iDNodoDestino = value; }
        }

        [DBField()]
        public DateTime Data
        {
            get { return data; }
            set { data = value; }
        }

        [DBField()]
        public long Contador
        {
            get { return contador; }
            set { contador = value; }
        }

    #endregion


    }
}
```

DBNodo

```
using DBManager.DBMapper;
using System.ComponentModel.DataAnnotations;

namespace Dealer
{
    [DBTable("dbo", "Nodo")]
    public class DBNodo
    {
    #region "DBNodo_Fields"
        private int? iD;
        private string endereco;
        private bool ativo;

    #endregion

    #region "DBNodo_Properties"

        [DBField()]
        [Key()]
        [Editable(false)]
        public int? ID
        {
            get { return iD; }
            set { iD = value; }
        }

        [DBField()]
        [StringLength(100, ErrorMessage="Property Endereco exceeded it's limits(0-100).")]
        public string Endereco
        {
            get { return endereco; }
            set { endereco = value; }
        }

        [DBField()]
        public bool Ativo
        {
            get { return ativo; }
            set { ativo = value; }
        }

    #endregion


    }
}
```

DBNumeracao

```
using DBManager.DBMapper;
using System.ComponentModel.DataAnnotations;

namespace Dealer
```

```
{
    [DBTable("dbo", "Numeracao")]
    public class DBNumeracao
    {
    #region "DBNumeracao_Fields"
        private int? iD;
        private long pedido;
        private long mensagem;

    #endregion

    #region "DBNumeracao_Properties"

        [DBField()]
        [Key()]
        [Editable(false)]
        public int? ID
        {
            get { return iD; }
            set { iD = value; }
        }

        [DBField()]
        public long Pedido
        {
            get { return pedido; }
            set { pedido = value; }
        }

        [DBField()]
        public long Mensagem
        {
            get { return mensagem; }
            set { mensagem = value; }
        }

    #endregion


    }
}
```

DBPedido

```
using DBManager.DBMapper;
using System.ComponentModel.DataAnnotations;
using System;

namespace Dealer
{
    [DBTable("dbo", "DBPedido")]
    public class DBPedido
    {
    #region "DBPedido_Fields"
```

```
        private long? iD;
        private long? iDCliente;
        private DateTime data;
        private int iDNodo;
        private long numero;

    #endregion

    #region "DBPedido_Properties"

        [DBField()]
        [Key()]
        [Editable(false)]
        public long? ID
        {
            get { return iD; }
            set { iD = value; }
        }

        [DBField()]
        public long? IDCliente
        {
            get { return iDCliente; }
            set { iDCliente = value; }
        }

        [DBField()]
        public DateTime Data
        {
            get { return data; }
            set { data = value; }
        }

        [DBField()]
        public int IDNodo
        {
            get { return iDNodo; }
            set { iDNodo = value; }
        }

        [DBField()]
        public long Numero
        {
            get { return numero; }
            set { numero = value; }
        }

    #endregion


    }
}
```

DBProduto

```
using DBManager.DBMapper;
using System.ComponentModel.DataAnnotations;

namespace Dealer
{
    [DBTable("dbo", "Produto")]
    public class DBProduto
    {
    #region "DBProduto_Fields"
        private long? iD;
        private string nome;
        private double preco;
        private string unidade;
        private double peso;
        private string observacoes;
        private string codigo;

    #endregion

    #region "DBProduto_Properties"

        [DBField()]
        [Key()]
        [Editable(false)]
        public long? ID
        {
            get { return iD; }
            set { iD = value; }
        }

        [DBField()]
        [StringLength(100, ErrorMessage="Property Nome exceeded it's limits(0-100).")]
        public string Nome
        {
            get { return nome; }
            set { nome = value; }
        }

        [DBField()]
        public double Preco
        {
            get { return preco; }
            set { preco = value; }
        }

        [DBField()]
        [StringLength(2, ErrorMessage="Property Unidade exceeded it's limits(0-2).")]
        public string Unidade
        {
            get { return unidade; }
            set { unidade = value; }
        }

        [DBField()]
        public double Peso
        {
```

```
            get { return peso; }
            set { peso = value; }
        }

        [DBField()]
        public string Observacoes
        {
            get { return observacoes; }
            set { observacoes = value; }
        }

        [DBField()]
        [DBAlternateKey()]
        public string Codigo
        {
            get { return codigo; }
            set { codigo = value; }
        }

    #endregion


    }
}
```

Correios.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Barsa;
using DBManager;
using DBManager.DBConnector;
using DBManager.DBMapper;
using DBManager.DBWarder;
using DBManager.DBImprover;

namespace Dealer.Classes
{

    //Padrões de Mensagens
    //Novo Produto -> NEWPRODUTO:<XML do objeto produto>
    //Adicionar Estoque -> ESTOQUE_INC:<S ou N>_QTD:<quantidade>_PROD:<IDProduto>
    //Novo Cliente -> NEWCLIENTE:<XML do cliente>
    //Solicitar Transf -> SOLTRANSF_PROD:<IDProduto>_QTD:<quantidade>
    //Transferir -> TRANSF_PROD:<IDProduto>_QTD:<quantidade>_IDREMET:<ID nodo
remetente>_IDDEST:<ID nodo destino>
    //Ack Transf -> ACKTRANSF:<ID Mensagem>


    public static class Correios
    {
        private static List<Nodo> nodosConhecidos = new List<Nodo>();
```

```
public static Nodo eu = new Nodo();
public static string Conexao = "DSN=DealerDataBase;OLE DB Services=-2;";

public static void Distribuir(String msg) {
  //TODO Enviar Mensagem para todos os nodos conhecidos
   Mensagem m = new Mensagem();
   m.Corpo = msg;
   m.Numero = PegarNrMsg(0);
   m.IDRemetente = eu.ID;
   ImpNodo iNd;
   int enviados = 0;

   foreach(Nodo n in nodosConhecidos){
      iNd = new ImpNodo();
      iNd.Converter(n);
      if (iNd.EnviarMensagem(m)) {
         enviados++;
      }
   }

}

private static long PegarNrMsg(int NrAtual){

   DBFactory f = new DBFactory(DBProvider.ODBC, Correios.Conexao);
   GenDAL gd = new GenDAL(f);

   List<DBNumeracao> lst = new List<DBNumeracao>();
   lst = gd.List<DBNumeracao>();
   DBNumeracao n = new DBNumeracao();

   long resul = 0;

   if (lst.Count > 0)
   {
      n.ID = lst.ElementAt(0).ID;
      if (gd.Open<DBNumeracao>(ref n)) {
         if (NrAtual > 0) {
            n.Mensagem = NrAtual;
         }
         n.Mensagem = n.Mensagem + 1;
         resul = n.Mensagem;
         gd.Save(ref n);
      }

   }
   else {
      n.ID = null;
      if(NrAtual > 0){
         n.Mensagem = NrAtual;
      }else{
         n.Mensagem = 1;
      }
      n.Pedido = 0;
      resul = 1;
      gd.Save(ref n);
```

```
            }


            return resul;
        }


    }
}
```

ImpCliente.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Barsa;
using DBManager;
using DBManager.DBConnector;
using DBManager.DBMapper;
using DBManager.DBWarder;
using DBManager.DBImprover;

namespace Dealer.Classes
{
    public class ImpCliente:Cliente
    {
        public int Cadastrar() {
            int i = 0;
            DBCliente dbC = new DBCliente();

            dbC.Bloqueado = this.Bloqueado;
            dbC.CodigoDeAcesso = this.CodigoDeAcesso;
            dbC.CodigoNacional = this.CodigoNacional;
            dbC.DataNascimento = this.DataNascimento;
            dbC.Email = this.EMail;
            dbC.ID = null;
            dbC.MsgBloqueado = this.MensagemBloqueio;
            dbC.Nome = this.Nome;

            DBFactory f = new DBFactory(DBProvider.ODBC, Correios.Conexao);
            GenDAL gd = new GenDAL(f);

            try
            {
                i = gd.Save(ref dbC);
            }
            catch (Exception err)
            {
                i = -1;
            }

            return i;
        }
```

```
public void Converter(Cliente c) {
    this.Bloqueado = c.Bloqueado;
    this.CodigoDeAcesso = c.CodigoDeAcesso;
    this.CodigoNacional = c.CodigoNacional;
    this.DataNascimento = c.DataNascimento;
    this.EMail = c.EMail;
    this.ID = c.ID;
    this.MensagemBloqueio = c.MensagemBloqueio;
    this.Nome = c.Nome;
}

public void Carregar(long id) {
    List<DBCliente> lst = new List<DBCliente>();

    DBFactory f = new DBFactory(DBProvider.ODBC, Correios.Conexao);
    GenDAL gd = new GenDAL(f);

    gd.AddFilter("ID", OperatorType.Equal, id);
    lst = gd.List<DBCliente>();

    if (lst.Count > 0)
    {
        this.Bloqueado = lst.ElementAt(0).Bloqueado;
        this.CodigoDeAcesso = lst.ElementAt(0).CodigoDeAcesso;
        this.CodigoNacional = lst.ElementAt(0).CodigoNacional;
        this.DataNascimento = lst.ElementAt(0).DataNascimento;
        this.EMail = lst.ElementAt(0).Email;
        this.ID = lst.ElementAt(0).ID;
        this.MensagemBloqueio = lst.ElementAt(0).MsgBloqueado;
        this.Nome = lst.ElementAt(0).Nome;

    }
}

public List<Cliente> Listar() {
    List<Cliente> res = new List<Cliente>();
    List<DBCliente> lst = new List<DBCliente>();
    Cliente cl;

    DBFactory f = new DBFactory(DBProvider.ODBC, Correios.Conexao);
    GenDAL gd = new GenDAL(f);

    if (this.Nome != "" && !(this.Nome == null))
    {
        gd.AddFilter("Nome", OperatorType.LikeStartsWith, this.Nome);
    }

    if (this.CodigoNacional != "" && !(this.CodigoNacional == null))
    {
        gd.AddFilter("CodigoNacional", OperatorType.Equal, this.CodigoNacional);
    }

    lst = gd.List<DBCliente>();

    foreach (DBCliente c in lst)
```

```
        {
            cl = new Cliente();
            cl.ID = c.ID;
            cl.Bloqueado = c.Bloqueado;
            cl.CodigoDeAcesso = c.CodigoDeAcesso;
            cl.CodigoNacional = c.CodigoNacional;
            cl.DataNascimento = c.DataNascimento;
            cl.EMail = c.Email;
            cl.MensagemBloqueio = c.MsgBloqueado;
            cl.Nome = c.Nome;

            res.Add(cl);
        }

        return res;
    }

  }
}
```

ImpCondicaoPagto.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Barsa;

namespace Dealer.Classes
{
   public class ImpCondicaoPagto:CondicaoPagto
   {
      public int NumeroParcelas
      {
         get
         {
            return this.Prazos.Count - 1;
         }
      }

      public bool AVista
      {
         get
         {
            if (this.NumeroParcelas == 1)
            {
               return true;
            }
            else
            {
               return false;
            }
         }
```

```
        }

        public bool TemEntrada
        {
            get
            {
                bool r = false;
                if (this.Prazos[0] == 0)
                {
                    r = true;
                }
                return r;
            }
        }

    }
}
```

ImpItem.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Barsa;

namespace Dealer
{
    public class ImpItem:Item
    {
    }
}
```

ImpNodo.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.ServiceModel.Channels;
using Barsa;
using DBManager;
using DBManager.DBConnector;
using DBManager.DBMapper;
using DBManager.DBWarder;
using DBManager.DBImprover;

namespace Dealer.Classes
{
    public class ImpNodo:Nodo
    {

        public void Converter(Nodo n) {
```

```
    this.Endereco = n.Endereco;
    this.ID = n.ID;
    this.Porta = n.Porta;
}

public bool EnviarMensagem(Mensagem m) {
    try {
        DealerWServ.DealerService w = new DealerWServ.DealerService();
        w.Url = "http://" + this.Endereco + ":" + this.Porta + "/DealerService.asmx";
        w.ReceberMensagem(Funcoes.Serializar(m));
    }
    catch (Exception err)
    {
        return false;
    }

    return true;

}

public void Cadastrar() {
    DBNodo dbN = new DBNodo();
    dbN.Ativo = true;
    dbN.Endereco = this.Endereco;
    dbN.ID = null;

    DBFactory f = new DBFactory(DBProvider.ODBC, Correios.Conexao);
    GenDAL gd = new GenDAL(f);
    int i;

    try
    {
        i = gd.Save(ref dbN);
    }
    catch (Exception err)
    {
        DBAKNotFoundException aknf = new DBAKNotFoundException("");
        if(err.GetType().Equals(aknf.GetType())){
            err = null;
        }
    }

    finally
    {
        if (gd != null)
        {
            gd.Dispose();
        }
    }


}
```

```
    }
}
```

ImpPedido.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Barsa;
using DBManager;
using DBManager.DBConnector;
using DBManager.DBMapper;
using DBManager.DBWarder;
using DBManager.DBImprover;

namespace Dealer.Classes
{
    public class ImpPedido:Pedido
    {
        public int Cadastrar() {
            int i = 0;
            DBPedido dbP = new DBPedido();
            DBItem dbI;

            dbP.Data = this.Data;
            dbP.ID = null;
            dbP.IDCliente = this.Cliente.ID;
            dbP.IDNodo = this.Nodo.ID;
            dbP.Numero = this.Numero;

            DBFactory f = new DBFactory(DBProvider.ODBC, Correios.Conexao);
            GenDAL gd = new GenDAL(f);

            try
            {
                i = gd.Save(ref dbP);

                foreach (Item it in this.Itens) {
                    dbI = new DBItem();
                    dbI.ID = null;
                    dbI.IDPedido = i;
                    dbI.IDProduto = it.Produto.ID;
                    dbI.Quantidade = it.Quantidade;
                    dbI.Valor = it.Valor;
                    dbI.ValorDesconto = it.Produto.Preco - it.Valor;

                    try{
                        i = gd.Save(ref dbP);
                    }catch (Exception err){

                    }
```

```
        }

        return i;
    }
    catch (Exception err) {
        return -1;
    }

}

public void Converter(Pedido p) {
    this.Cliente = p.Cliente;
    this.Data = p.Data;
    this.ID = p.ID;
    this.Itens = p.Itens;
    this.Nodo = p.Nodo;
    this.Numero = p.Numero;
}

public void Carregar(long id) {
    List<DBPedido> lst = new List<DBPedido>();
    List<DBItem> lstI = new List<DBItem>();
    Item it;

    DBFactory f = new DBFactory(DBProvider.ODBC, Correios.Conexao);
    GenDAL gd = new GenDAL(f);

    gd.AddFilter("ID", OperatorType.Equal, id);
    lst = gd.List<DBPedido>();
    gd.Clear();
    gd.AddFilter("Pedido", OperatorType.Equal, id);
    lstI = gd.List<DBItem>();

    if (lst.Count > 0)
    {
        this.Cliente = new Cliente();
        this.Cliente.ID  = lst.ElementAt(0).IDCliente;
        this.Data = lst.ElementAt(0).Data;
        this.ID = lst.ElementAt(0).ID;
        this.Itens = new List<Item>();
        foreach (DBItem i in lstI)
        {
            it = new Item();
            it.ID = i.ID;
            it.Produto = new Produto();
            it.Produto.ID = i.IDProduto;
            it.Quantidade = i.Quantidade;
            it.Valor = i.Valor;
            this.Itens.Add(it);
        }
        this.Nodo = new Nodo();
        this.Nodo.ID = lst.ElementAt(0).IDNodo;
        this.Numero = lst.ElementAt(0).Numero;

    }
```

```
}

public List<Pedido> Listar() {
    List<Pedido> res = new List<Pedido>();
    List<DBPedido> lst = new List<DBPedido>();
    Pedido ped;

    DBFactory f = new DBFactory(DBProvider.ODBC, Correios.Conexao);
    GenDAL gd = new GenDAL(f);

    if (!(this.Numero == null))
    {
        gd.AddFilter("Numero", OperatorType.LikeStartsWith, this.Numero);
    }


    lst = gd.List<DBPedido>();

    foreach (DBPedido p in lst)
    {
        ped = new Pedido();
        ped.ID = p.ID;
        ped.Cliente = new Cliente();
        ped.Cliente.ID = p.IDCliente;
        ped.Data = p.Data;
        ped.Itens = new List<Item>();
        ped.Nodo = new Nodo();
        ped.Nodo.ID = p.IDNodo;
        ped.Numero = p.Numero;

        res.Add(ped);
    }

    return res;
}

public long PegarNumeroPedido() {
    DBFactory f = new DBFactory(DBProvider.ODBC, Correios.Conexao);
    GenDAL gd = new GenDAL(f);

    List<DBNumeracao> lst = new List<DBNumeracao>();
    lst = gd.List<DBNumeracao>();
    DBNumeracao n = new DBNumeracao();

    long resul = 0;

    if (lst.Count > 0)
    {
        n.ID = lst.ElementAt(0).ID;
        if (gd.Open<DBNumeracao>(ref n))
        {
            n.Pedido = n.Pedido + 1;
            resul = n.Pedido;
            gd.Save(ref n);
        }
```

```
            }
            else
            {
                n.ID = null;
                n.Pedido = 1;
                n.Mensagem = 0;
                resul = 1;
                gd.Save(ref n);
            }


            return resul;
        }
    }
}
```

ImpProduto.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Barsa;
using DBManager;
using DBManager.DBConnector;
using DBManager.DBMapper;
using DBManager.DBWarder;
using DBManager.DBImprover;

namespace Dealer.Classes
{
    public class ImpProduto:Produto
    {
        public int Cadastrar() {
            int i = 0;
            DBProduto dbP = new DBProduto();

            dbP.ID = null;
            dbP.Nome = this.Nome;
            dbP.Observacoes = this.Observacoes;
            dbP.Peso = this.Peso;
            dbP.Preco = this.Preco;
            dbP.Unidade = this.Unidade;
            dbP.Codigo = this.Codigo;

            DBFactory f = new DBFactory(DBProvider.ODBC, Correios.Conexao);
            GenDAL gd = new GenDAL(f);

            try {
                i = gd.Save(ref dbP);
            }
            catch (Exception err)
            {
```

```
            i = -1;
        }

        return i;
    }

    public void Converter(Produto p) {
        this.Codigo = p.Codigo;
        this.ID = p.ID;
        this.Nome = p.Nome;
        this.Observacoes = p.Observacoes;
        this.Peso = p.Peso;
        this.Preco = p.Preco;
        this.Promocao = p.Promocao;
        this.Unidade = p.Unidade;
    }

    public void Carregar(long id) {
        List<DBProduto> lst = new List<DBProduto>();

        DBFactory f = new DBFactory(DBProvider.ODBC, Correios.Conexao);
        GenDAL gd = new GenDAL(f);

        gd.AddFilter("ID", OperatorType.Equal, id);
        lst = gd.List<DBProduto>();

        if (lst.Count > 0) {
            this.Codigo = lst.ElementAt(0).Codigo;
            this.ID = id;
            this.Nome = lst.ElementAt(0).Nome;
            this.Observacoes = lst.ElementAt(0).Observacoes;
            this.Peso = lst.ElementAt(0).Peso;
            this.Preco = lst.ElementAt(0).Preco;
            //this.Promocao = lst.ElementAt(0).
            this.Unidade = lst.ElementAt(0).Unidade;
        }

    }

    public List<Produto> Listar() {
        List<Produto> res = new List<Produto>();
        List<DBProduto> lst = new List<DBProduto>();
        Produto pr;

        DBFactory f = new DBFactory(DBProvider.ODBC, Correios.Conexao);
        GenDAL gd = new GenDAL(f);

        if(this.Nome != "" && ! (this.Nome == null)){
            gd.AddFilter("Nome", OperatorType.LikeStartsWith, this.Nome);
        }

        if (this.Codigo != "" && !(this.Codigo == null))
        {
            gd.AddFilter("Codigo", OperatorType.Equal, this.Codigo);
        }
```

```
            lst = gd.List<DBProduto>();

            foreach (DBProduto p in lst) {
                pr = new Produto();
                pr.ID = p.ID;
                //pr.Codigo =
                pr.Nome = p.Nome;
                pr.Observacoes = p.Observacoes;
                pr.Peso = p.Peso;
                pr.Preco = p.Preco;
                pr.Unidade = p.Unidade;
                pr.Codigo = p.Codigo;
                res.Add(pr);
            }

            return res;
        }

    }
}
```

ImpVendedor.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using Barsa;

namespace Dealer.Classes
{
    public class ImpVendedor:Vendedor
    {
    }
}
```

DealerService.asmx

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;
using Dealer.Classes;
using Barsa;

namespace Dealer
{
    /// <summary>
    /// Summary description for DealerService
    /// </summary>
    [WebService(Namespace = "http://tempuri.org/")]
    [WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1_1)]
```

```
    [System.ComponentModel.ToolboxItem(false)]
    // To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.
    // [System.Web.Script.Services.ScriptService]
    public class DealerService : System.Web.Services.WebService
    {

        [WebMethod]
        public string HelloWorld()
        {
            return "Hello World";
        }

        [WebMethod]
        public string TesteBD() {
            return "Hello";
        }

        [WebMethod]
        public string FazerPedido(string XMLPedido){
            Pedido p = new Pedido();
            p = Funcoes.Deserializar<Pedido>(XMLPedido);

            if(p == null){
                return "";
            }

            ImpPedido iPed = new ImpPedido();
            iPed.Converter(p);

            return iPed.Cadastrar() + "";
        }

        [WebMethod]
        public string ObterPedidos(string XMLPedidoParametro) {
            Pedido p = new Pedido();
            p = Funcoes.Deserializar<Pedido>(XMLPedidoParametro);

            if (p == null)
            {
                return "";
            }

            ImpPedido iPed = new ImpPedido();
            iPed.Converter(p);
            List<Pedido> lst = iPed.Listar();

            if (lst == null)
            {
                return "";
            }


            return Funcoes.Serializar(lst);

        }
```

```
        [WebMethod]
        public Boolean AdicionarNodo(string XMLNodo) {
            Nodo n = new Nodo();
            n = Funcoes.Deserializar<Nodo>(XMLNodo);

            if(n == null){
                return false;
            }

            ImpNodo iNd = new ImpNodo();
            iNd.Converter(n);
            iNd.Cadastrar();

            return true;
        }

        [WebMethod]
        public Boolean AdicionarEstoque(long IDProduto, double Quantidade, string NrDocto, string Motivo,
bool Incremento) {
            string increm = "";
            string m = "";

            if (Incremento)
            {
                increm = "S";
            }
            else {
                increm = "N";
            }

            m = "ESTOQUE_INC:" + increm + "_QTD:" + Quantidade + "_PROD:" + IDProduto;
            Correios.Distribuir(m);

            return true;
        }

        [WebMethod]
        public Boolean AdicionarCliente(string XMLCliente) {
            Cliente c = Funcoes.Deserializar<Cliente>(XMLCliente);
            ImpCliente iC = new ImpCliente();
            iC.Converter(c);
            if (iC.Cadastrar() == 1)
            {
                return true;
            }
            else
            {
                return false;
            }
        }

        [WebMethod]
        public string AdicionarProduto(string XMLProduto) {
            Produto p = Funcoes.Deserializar<Produto>(XMLProduto);
            ImpProduto iP = new ImpProduto();
```

```
            iP.Converter(p);
            if (iP.Cadastrar() == 1)
            {
                return "OK";
            }
            else
            {
                return "N";
            }
        }

[WebMethod]
        public string ReceberMensagem(string XMLMensagem) {
            Mensagem m = Funcoes.Deserializar<Mensagem>(XMLMensagem);
            return Correios.ReceberMensagem(m);
        }
```

Web.config

```
<?xml version="1.0"?>
<configuration>
            <configSections>
  <sectionGroup      name="applicationSettings"      type="System.Configuration.ApplicationSettingsGroup,
System, Version=4.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089" >
    <section       name="Dealer.Properties.Settings"       type="System.Configuration.ClientSettingsSection,
System,          Version=4.0.0.0,          Culture=neutral,          PublicKeyToken=b77a5c561934e089"
requirePermission="false" />
  </sectionGroup>
 </configSections>
 <appSettings/>
            <connectionStrings/>
            <system.web>
                        <compilation debug="true" targetFramework="4.0">
                        </compilation>
                        <!--
     The <authentication> section enables configuration
     of the security authentication mode used by
     ASP.NET to identify an incoming user.
   -->
                        <authentication mode="Windows"/>
                        <!--
      The <customErrors> section enables configuration
      of what to do if/when an unhandled error occurs
      during the execution of a request. Specifically,
      it enables developers to configure html error pages
      to be displayed in place of a error stack trace.

      <customErrors mode="RemoteOnly" defaultRedirect="GenericErrorPage.htm">
        <error statusCode="403" redirect="NoAccess.htm" />
        <error statusCode="404" redirect="FileNotFound.htm" />
      </customErrors>
   -->
                        <pages                                                controlRenderingCompatibilityVersion="3.5"
clientIDMode="AutoID"/></system.web>
```

```
        <!--
      The system.webServer section is required for running ASP.NET AJAX under Internet
      Information Services 7.0.  It is not necessary for previous version of IIS.
   -->
 <applicationSettings>
  <Dealer.Properties.Settings>
   <setting name="Dealer_DealerWServ_DealerService" serializeAs="String">
    <value>http://localhost:52495/DealerService.asmx</value>
   </setting>
  </Dealer.Properties.Settings>
 </applicationSettings>
</configuration>
```

Sherlock

Service1.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Diagnostics;
using System.Linq;
using System.ServiceProcess;
using System.Text;

namespace Sherlock
{
    public partial class Service1 : ServiceBase
    {
        private Sherlock shlk;
        public Service1()
        {
            InitializeComponent();
        }

        protected override void OnStart(string[] args)
        {
            timer1.Interval = 1000;
            timer1.Enabled = true;
            shlk = new Sherlock();
            shlk.Iniciar();
        }

        protected override void OnStop()
        {
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            shlk.VerificarNodo();
        }
    }
```

```
}
```

Sherlock.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Barsa;

namespace Sherlock
{
   class  Sherlock
   {
      private List<Nodo> nodos;
      private int nrNodo;

      public void Iniciar() {
         nodos = new List<Nodo>();
         nrNodo = 0;
      }

      public void VerificarNodo() {
         DealerWServ.DealerServiceSoapClient w = new DealerWServ.DealerServiceSoapClient();
         Mensagem m = new Mensagem();
         m.Corpo = "ACKNODO:" + nodos[nrNodo];
         string resp = w.ReceberMensagem(Funcoes.Serializar(m));

         if (resp == "-1")
         {
            NodoDesativado(nodos[nrNodo]);
         }
         else {
            Nodo n = Funcoes.Deserializar<Nodo>(resp);
            if(!(n is  DBNull)){
               AdicionarNodo(n);
            }
         }
         nrNodo++;
      }
   }
}
```

Cliente Windows

frmPrincipal.design

```
namespace ExemploCliente
{
   partial class frmPrincipal
   {
      /// <summary>
```

```
        /// Required designer variable.
        /// </summary>
        private System.ComponentModel.IContainer components = null;

        /// <summary>
        /// Clean up any resources being used.
        /// </summary>
        /// <param name="disposing">true if managed resources should be disposed; otherwise,
false.</param>
        protected override void Dispose(bool disposing)
        {
            if (disposing && (components != null))
            {
                components.Dispose();
            }
            base.Dispose(disposing);
        }

        #region Windows Form Designer generated code

        /// <summary>
        /// Required method for Designer support - do not modify
        /// the contents of this method with the code editor.
        /// </summary>
        private void InitializeComponent()
        {
            this.groupBox1 = new System.Windows.Forms.GroupBox();
            this.txtPorta = new System.Windows.Forms.TextBox();
            this.txtEndereco = new System.Windows.Forms.TextBox();
            this.btnAdicionar = new System.Windows.Forms.Button();
            this.btnTestar = new System.Windows.Forms.Button();
            this.label2 = new System.Windows.Forms.Label();
            this.label1 = new System.Windows.Forms.Label();
            this.tabControl1 = new System.Windows.Forms.TabControl();
            this.tabPage1 = new System.Windows.Forms.TabPage();
            this.btnProdAdd = new System.Windows.Forms.Button();
            this.label8 = new System.Windows.Forms.Label();
            this.txtProdObs = new System.Windows.Forms.TextBox();
            this.txtProdPeso = new System.Windows.Forms.TextBox();
            this.label7 = new System.Windows.Forms.Label();
            this.label6 = new System.Windows.Forms.Label();
            this.label5 = new System.Windows.Forms.Label();
            this.txtProdUnid = new System.Windows.Forms.TextBox();
            this.txtProdValor = new System.Windows.Forms.TextBox();
            this.label4 = new System.Windows.Forms.Label();
            this.txtProdNome = new System.Windows.Forms.TextBox();
            this.txtProdCod = new System.Windows.Forms.TextBox();
            this.label3 = new System.Windows.Forms.Label();
            this.tabPage2 = new System.Windows.Forms.TabPage();
            this.btnCliAdd = new System.Windows.Forms.Button();
            this.txtCliNasc = new System.Windows.Forms.MaskedTextBox();
            this.label13 = new System.Windows.Forms.Label();
            this.label12 = new System.Windows.Forms.Label();
            this.txtCliSenha = new System.Windows.Forms.TextBox();
            this.label11 = new System.Windows.Forms.Label();
            this.txtCliEmail = new System.Windows.Forms.TextBox();
```

```
this.label10 = new System.Windows.Forms.Label();
this.label9 = new System.Windows.Forms.Label();
this.txtCliNome = new System.Windows.Forms.TextBox();
this.txtCliCodNac = new System.Windows.Forms.TextBox();
this.tabPage3 = new System.Windows.Forms.TabPage();
this.btnEstqSet = new System.Windows.Forms.Button();
this.btnEstqInc = new System.Windows.Forms.Button();
this.txtEstqQtd = new System.Windows.Forms.NumericUpDown();
this.label15 = new System.Windows.Forms.Label();
this.txtEstqCod = new System.Windows.Forms.TextBox();
this.label14 = new System.Windows.Forms.Label();
this.tabPage4 = new System.Windows.Forms.TabPage();
this.btnPedAdicionar = new System.Windows.Forms.Button();
this.btnPedAdd = new System.Windows.Forms.Button();
this.label21 = new System.Windows.Forms.Label();
this.txtPedValor = new System.Windows.Forms.NumericUpDown();
this.txtPedQtd = new System.Windows.Forms.NumericUpDown();
this.label20 = new System.Windows.Forms.Label();
this.label19 = new System.Windows.Forms.Label();
this.txtPedProd = new System.Windows.Forms.TextBox();
this.label18 = new System.Windows.Forms.Label();
this.lstPedItens = new System.Windows.Forms.ListView();
this.txtPedCli = new System.Windows.Forms.TextBox();
this.label17 = new System.Windows.Forms.Label();
this.txtPedData = new System.Windows.Forms.MaskedTextBox();
this.label16 = new System.Windows.Forms.Label();
this.tabPage5 = new System.Windows.Forms.TabPage();
this.label22 = new System.Windows.Forms.Label();
this.listView1 = new System.Windows.Forms.ListView();
this.btnConsPedFiltr = new System.Windows.Forms.Button();
this.txtConsPedData = new System.Windows.Forms.MaskedTextBox();
this.tabPage6 = new System.Windows.Forms.TabPage();
this.btnConsCliFiltr = new System.Windows.Forms.Button();
this.listView2 = new System.Windows.Forms.ListView();
this.groupBox2 = new System.Windows.Forms.GroupBox();
this.lstNodos = new System.Windows.Forms.ListBox();
this.groupBox1.SuspendLayout();
this.tabControl1.SuspendLayout();
this.tabPage1.SuspendLayout();
this.tabPage2.SuspendLayout();
this.tabPage3.SuspendLayout();
((System.ComponentModel.ISupportInitialize)(this.txtEstqQtd)).BeginInit();
this.tabPage4.SuspendLayout();
((System.ComponentModel.ISupportInitialize)(this.txtPedValor)).BeginInit();
((System.ComponentModel.ISupportInitialize)(this.txtPedQtd)).BeginInit();
this.tabPage5.SuspendLayout();
this.tabPage6.SuspendLayout();
this.groupBox2.SuspendLayout();
this.SuspendLayout();
//
// groupBox1
//
this.groupBox1.Controls.Add(this.txtPorta);
this.groupBox1.Controls.Add(this.txtEndereco);
this.groupBox1.Controls.Add(this.btnAdicionar);
this.groupBox1.Controls.Add(this.btnTestar);
```

```
this.groupBox1.Controls.Add(this.label2);
this.groupBox1.Controls.Add(this.label1);
this.groupBox1.Location = new System.Drawing.Point(11, 12);
this.groupBox1.Name = "groupBox1";
this.groupBox1.Size = new System.Drawing.Size(350, 89);
this.groupBox1.TabIndex = 0;
this.groupBox1.TabStop = false;
this.groupBox1.Text = "Adicionar um Nodo";
//
// txtPorta
//
this.txtPorta.Location = new System.Drawing.Point(65, 54);
this.txtPorta.Name = "txtPorta";
this.txtPorta.Size = new System.Drawing.Size(69, 20);
this.txtPorta.TabIndex = 5;
//
// txtEndereco
//
this.txtEndereco.Location = new System.Drawing.Point(65, 28);
this.txtEndereco.Name = "txtEndereco";
this.txtEndereco.Size = new System.Drawing.Size(277, 20);
this.txtEndereco.TabIndex = 4;
//
// btnAdicionar
//
this.btnAdicionar.Location = new System.Drawing.Point(255, 54);
this.btnAdicionar.Name = "btnAdicionar";
this.btnAdicionar.Size = new System.Drawing.Size(89, 29);
this.btnAdicionar.TabIndex = 3;
this.btnAdicionar.Text = "Adicionar";
this.btnAdicionar.UseVisualStyleBackColor = true;
this.btnAdicionar.Click += new System.EventHandler(this.btnAdicionar_Click);
//
// btnTestar
//
this.btnTestar.Location = new System.Drawing.Point(161, 54);
this.btnTestar.Name = "btnTestar";
this.btnTestar.Size = new System.Drawing.Size(88, 29);
this.btnTestar.TabIndex = 2;
this.btnTestar.Text = "Testar";
this.btnTestar.UseVisualStyleBackColor = true;
this.btnTestar.Click += new System.EventHandler(this.btnTestar_Click);
//
// label2
//
this.label2.AutoSize = true;
this.label2.Location = new System.Drawing.Point(6, 57);
this.label2.Name = "label2";
this.label2.Size = new System.Drawing.Size(32, 13);
this.label2.TabIndex = 1;
this.label2.Text = "Porta";
//
// label1
//
this.label1.AutoSize = true;
this.label1.Location = new System.Drawing.Point(6, 31);
```

```
this.label1.Name = "label1";
this.label1.Size = new System.Drawing.Size(53, 13);
this.label1.TabIndex = 0;
this.label1.Text = "Endereço";
//
// tabControl1
//
this.tabControl1.Controls.Add(this.tabPage1);
this.tabControl1.Controls.Add(this.tabPage2);
this.tabControl1.Controls.Add(this.tabPage3);
this.tabControl1.Controls.Add(this.tabPage4);
this.tabControl1.Controls.Add(this.tabPage5);
this.tabControl1.Controls.Add(this.tabPage6);
this.tabControl1.Location = new System.Drawing.Point(12, 111);
this.tabControl1.Name = "tabControl1";
this.tabControl1.SelectedIndex = 0;
this.tabControl1.Size = new System.Drawing.Size(437, 292);
this.tabControl1.TabIndex = 1;
//
// tabPage1
//
this.tabPage1.Controls.Add(this.btnProdAdd);
this.tabPage1.Controls.Add(this.label8);
this.tabPage1.Controls.Add(this.txtProdObs);
this.tabPage1.Controls.Add(this.txtProdPeso);
this.tabPage1.Controls.Add(this.label7);
this.tabPage1.Controls.Add(this.label6);
this.tabPage1.Controls.Add(this.label5);
this.tabPage1.Controls.Add(this.txtProdUnid);
this.tabPage1.Controls.Add(this.txtProdValor);
this.tabPage1.Controls.Add(this.label4);
this.tabPage1.Controls.Add(this.txtProdNome);
this.tabPage1.Controls.Add(this.txtProdCod);
this.tabPage1.Controls.Add(this.label3);
this.tabPage1.Location = new System.Drawing.Point(4, 22);
this.tabPage1.Name = "tabPage1";
this.tabPage1.Padding = new System.Windows.Forms.Padding(3);
this.tabPage1.Size = new System.Drawing.Size(429, 266);
this.tabPage1.TabIndex = 0;
this.tabPage1.Text = "Produto";
this.tabPage1.UseVisualStyleBackColor = true;
//
// btnProdAdd
//
this.btnProdAdd.Location = new System.Drawing.Point(334, 226);
this.btnProdAdd.Name = "btnProdAdd";
this.btnProdAdd.Size = new System.Drawing.Size(89, 29);
this.btnProdAdd.TabIndex = 13;
this.btnProdAdd.Text = "Adicionar";
this.btnProdAdd.UseVisualStyleBackColor = true;
this.btnProdAdd.Click += new System.EventHandler(this.btnProdAdd_Click);
//
// label8
//
this.label8.AutoSize = true;
this.label8.Location = new System.Drawing.Point(6, 110);
```

```
this.label8.Name = "label8";
this.label8.Size = new System.Drawing.Size(44, 13);
this.label8.TabIndex = 12;
this.label8.Text = "Observ.";
//
// txtProdObs
//
this.txtProdObs.Location = new System.Drawing.Point(60, 110);
this.txtProdObs.Multiline = true;
this.txtProdObs.Name = "txtProdObs";
this.txtProdObs.Size = new System.Drawing.Size(363, 111);
this.txtProdObs.TabIndex = 11;
//
// txtProdPeso
//
this.txtProdPeso.Location = new System.Drawing.Point(60, 84);
this.txtProdPeso.Name = "txtProdPeso";
this.txtProdPeso.Size = new System.Drawing.Size(96, 20);
this.txtProdPeso.TabIndex = 9;
//
// label7
//
this.label7.AutoSize = true;
this.label7.Location = new System.Drawing.Point(6, 87);
this.label7.Name = "label7";
this.label7.Size = new System.Drawing.Size(31, 13);
this.label7.TabIndex = 8;
this.label7.Text = "Peso";
//
// label6
//
this.label6.AutoSize = true;
this.label6.Location = new System.Drawing.Point(296, 57);
this.label6.Name = "label6";
this.label6.Size = new System.Drawing.Size(47, 13);
this.label6.TabIndex = 7;
this.label6.Text = "Unidade";
//
// label5
//
this.label5.AutoSize = true;
this.label5.Location = new System.Drawing.Point(6, 61);
this.label5.Name = "label5";
this.label5.Size = new System.Drawing.Size(31, 13);
this.label5.TabIndex = 6;
this.label5.Text = "Valor";
//
// txtProdUnid
//
this.txtProdUnid.Location = new System.Drawing.Point(349, 54);
this.txtProdUnid.Name = "txtProdUnid";
this.txtProdUnid.Size = new System.Drawing.Size(74, 20);
this.txtProdUnid.TabIndex = 5;
//
// txtProdValor
//
```

```
this.txtProdValor.Location = new System.Drawing.Point(60, 58);
this.txtProdValor.Name = "txtProdValor";
this.txtProdValor.Size = new System.Drawing.Size(96, 20);
this.txtProdValor.TabIndex = 4;
//
// label4
//
this.label4.AutoSize = true;
this.label4.Location = new System.Drawing.Point(6, 35);
this.label4.Name = "label4";
this.label4.Size = new System.Drawing.Size(35, 13);
this.label4.TabIndex = 3;
this.label4.Text = "Nome";
//
// txtProdNome
//
this.txtProdNome.Location = new System.Drawing.Point(60, 32);
this.txtProdNome.Name = "txtProdNome";
this.txtProdNome.Size = new System.Drawing.Size(363, 20);
this.txtProdNome.TabIndex = 2;
//
// txtProdCod
//
this.txtProdCod.Location = new System.Drawing.Point(60, 6);
this.txtProdCod.Name = "txtProdCod";
this.txtProdCod.Size = new System.Drawing.Size(96, 20);
this.txtProdCod.TabIndex = 1;
//
// label3
//
this.label3.AutoSize = true;
this.label3.Location = new System.Drawing.Point(6, 9);
this.label3.Name = "label3";
this.label3.Size = new System.Drawing.Size(40, 13);
this.label3.TabIndex = 0;
this.label3.Text = "Código";
//
// tabPage2
//
this.tabPage2.Controls.Add(this.btnCliAdd);
this.tabPage2.Controls.Add(this.txtCliNasc);
this.tabPage2.Controls.Add(this.label13);
this.tabPage2.Controls.Add(this.label12);
this.tabPage2.Controls.Add(this.txtCliSenha);
this.tabPage2.Controls.Add(this.label11);
this.tabPage2.Controls.Add(this.txtCliEmail);
this.tabPage2.Controls.Add(this.label10);
this.tabPage2.Controls.Add(this.label9);
this.tabPage2.Controls.Add(this.txtCliNome);
this.tabPage2.Controls.Add(this.txtCliCodNac);
this.tabPage2.Location = new System.Drawing.Point(4, 22);
this.tabPage2.Name = "tabPage2";
this.tabPage2.Padding = new System.Windows.Forms.Padding(3);
this.tabPage2.Size = new System.Drawing.Size(429, 266);
this.tabPage2.TabIndex = 1;
this.tabPage2.Text = "Cliente";
```

```
this.tabPage2.UseVisualStyleBackColor = true;
//
// btnCliAdd
//
this.btnCliAdd.Location = new System.Drawing.Point(334, 226);
this.btnCliAdd.Name = "btnCliAdd";
this.btnCliAdd.Size = new System.Drawing.Size(89, 29);
this.btnCliAdd.TabIndex = 14;
this.btnCliAdd.Text = "Adicionar";
this.btnCliAdd.UseVisualStyleBackColor = true;
this.btnCliAdd.Click += new System.EventHandler(this.btnCliAdd_Click);
//
// txtCliNasc
//
this.txtCliNasc.Location = new System.Drawing.Point(74, 110);
this.txtCliNasc.Mask = "00/00/0000";
this.txtCliNasc.Name = "txtCliNasc";
this.txtCliNasc.Size = new System.Drawing.Size(71, 20);
this.txtCliNasc.TabIndex = 9;
this.txtCliNasc.ValidatingType = typeof(System.DateTime);
//
// label13
//
this.label13.AutoSize = true;
this.label13.Location = new System.Drawing.Point(6, 113);
this.label13.Name = "label13";
this.label13.Size = new System.Drawing.Size(63, 13);
this.label13.TabIndex = 8;
this.label13.Text = "Nascimento";
//
// label12
//
this.label12.AutoSize = true;
this.label12.Location = new System.Drawing.Point(6, 87);
this.label12.Name = "label12";
this.label12.Size = new System.Drawing.Size(38, 13);
this.label12.TabIndex = 7;
this.label12.Text = "Senha";
//
// txtCliSenha
//
this.txtCliSenha.Location = new System.Drawing.Point(74, 84);
this.txtCliSenha.Name = "txtCliSenha";
this.txtCliSenha.PasswordChar = '*';
this.txtCliSenha.Size = new System.Drawing.Size(184, 20);
this.txtCliSenha.TabIndex = 6;
this.txtCliSenha.UseSystemPasswordChar = true;
//
// label11
//
this.label11.AutoSize = true;
this.label11.Location = new System.Drawing.Point(6, 61);
this.label11.Name = "label11";
this.label11.Size = new System.Drawing.Size(35, 13);
this.label11.TabIndex = 5;
this.label11.Text = "E-mail";
```

```
//
// txtCliEmail
//
this.txtCliEmail.Location = new System.Drawing.Point(74, 58);
this.txtCliEmail.Name = "txtCliEmail";
this.txtCliEmail.Size = new System.Drawing.Size(258, 20);
this.txtCliEmail.TabIndex = 4;
//
// label10
//
this.label10.AutoSize = true;
this.label10.Location = new System.Drawing.Point(6, 35);
this.label10.Name = "label10";
this.label10.Size = new System.Drawing.Size(35, 13);
this.label10.TabIndex = 3;
this.label10.Text = "Nome";
//
// label9
//
this.label9.AutoSize = true;
this.label9.Location = new System.Drawing.Point(6, 9);
this.label9.Name = "label9";
this.label9.Size = new System.Drawing.Size(59, 13);
this.label9.TabIndex = 2;
this.label9.Text = "CPF/CNPJ";
//
// txtCliNome
//
this.txtCliNome.Location = new System.Drawing.Point(74, 32);
this.txtCliNome.Name = "txtCliNome";
this.txtCliNome.Size = new System.Drawing.Size(260, 20);
this.txtCliNome.TabIndex = 1;
//
// txtCliCodNac
//
this.txtCliCodNac.Location = new System.Drawing.Point(74, 6);
this.txtCliCodNac.Name = "txtCliCodNac";
this.txtCliCodNac.Size = new System.Drawing.Size(184, 20);
this.txtCliCodNac.TabIndex = 0;
//
// tabPage3
//
this.tabPage3.Controls.Add(this.btnEstqSet);
this.tabPage3.Controls.Add(this.btnEstqInc);
this.tabPage3.Controls.Add(this.txtEstqQtd);
this.tabPage3.Controls.Add(this.label15);
this.tabPage3.Controls.Add(this.txtEstqCod);
this.tabPage3.Controls.Add(this.label14);
this.tabPage3.Location = new System.Drawing.Point(4, 22);
this.tabPage3.Name = "tabPage3";
this.tabPage3.Size = new System.Drawing.Size(429, 266);
this.tabPage3.TabIndex = 2;
this.tabPage3.Text = "Estoque";
this.tabPage3.UseVisualStyleBackColor = true;
//
// btnEstqSet
```

```
//
this.btnEstqSet.Location = new System.Drawing.Point(242, 226);
this.btnEstqSet.Name = "btnEstqSet";
this.btnEstqSet.Size = new System.Drawing.Size(89, 29);
this.btnEstqSet.TabIndex = 5;
this.btnEstqSet.Text = "Lançar";
this.btnEstqSet.UseVisualStyleBackColor = true;
//
// btnEstqInc
//
this.btnEstqInc.Location = new System.Drawing.Point(337, 226);
this.btnEstqInc.Name = "btnEstqInc";
this.btnEstqInc.Size = new System.Drawing.Size(89, 29);
this.btnEstqInc.TabIndex = 4;
this.btnEstqInc.Text = "Incrementar";
this.btnEstqInc.UseVisualStyleBackColor = true;
//
// txtEstqQtd
//
this.txtEstqQtd.DecimalPlaces = 2;
this.txtEstqQtd.Location = new System.Drawing.Point(75, 32);
this.txtEstqQtd.Maximum = new decimal(new int[] {
-1530494976,
232830,
0,
0});
this.txtEstqQtd.Minimum = new decimal(new int[] {
-727379968,
232,
0,
-2147483648});
this.txtEstqQtd.Name = "txtEstqQtd";
this.txtEstqQtd.Size = new System.Drawing.Size(94, 20);
this.txtEstqQtd.TabIndex = 3;
//
// label15
//
this.label15.AutoSize = true;
this.label15.Location = new System.Drawing.Point(3, 34);
this.label15.Name = "label15";
this.label15.Size = new System.Drawing.Size(62, 13);
this.label15.TabIndex = 2;
this.label15.Text = "Quantidade";
//
// txtEstqCod
//
this.txtEstqCod.Location = new System.Drawing.Point(75, 6);
this.txtEstqCod.Name = "txtEstqCod";
this.txtEstqCod.Size = new System.Drawing.Size(94, 20);
this.txtEstqCod.TabIndex = 1;
//
// label14
//
this.label14.AutoSize = true;
this.label14.Location = new System.Drawing.Point(3, 9);
this.label14.Name = "label14";
```

```
this.label14.Size = new System.Drawing.Size(66, 13);
this.label14.TabIndex = 0;
this.label14.Text = "Cód.Produto";
//
// tabPage4
//
this.tabPage4.Controls.Add(this.btnPedAdicionar);
this.tabPage4.Controls.Add(this.btnPedAdd);
this.tabPage4.Controls.Add(this.label21);
this.tabPage4.Controls.Add(this.txtPedValor);
this.tabPage4.Controls.Add(this.txtPedQtd);
this.tabPage4.Controls.Add(this.label20);
this.tabPage4.Controls.Add(this.label19);
this.tabPage4.Controls.Add(this.txtPedProd);
this.tabPage4.Controls.Add(this.label18);
this.tabPage4.Controls.Add(this.lstPedItens);
this.tabPage4.Controls.Add(this.txtPedCli);
this.tabPage4.Controls.Add(this.label17);
this.tabPage4.Controls.Add(this.txtPedData);
this.tabPage4.Controls.Add(this.label16);
this.tabPage4.Location = new System.Drawing.Point(4, 22);
this.tabPage4.Name = "tabPage4";
this.tabPage4.Size = new System.Drawing.Size(429, 266);
this.tabPage4.TabIndex = 3;
this.tabPage4.Text = "Pedido";
this.tabPage4.UseVisualStyleBackColor = true;
//
// btnPedAdicionar
//
this.btnPedAdicionar.Location = new System.Drawing.Point(337, 224);
this.btnPedAdicionar.Name = "btnPedAdicionar";
this.btnPedAdicionar.Size = new System.Drawing.Size(89, 29);
this.btnPedAdicionar.TabIndex = 22;
this.btnPedAdicionar.Text = "Adicionar";
this.btnPedAdicionar.UseVisualStyleBackColor = true;
this.btnPedAdicionar.Click += new System.EventHandler(this.btnPedAdicionar_Click);
//
// btnPedAdd
//
this.btnPedAdd.Location = new System.Drawing.Point(386, 71);
this.btnPedAdd.Name = "btnPedAdd";
this.btnPedAdd.Size = new System.Drawing.Size(40, 23);
this.btnPedAdd.TabIndex = 21;
this.btnPedAdd.Text = "+";
this.btnPedAdd.UseVisualStyleBackColor = true;
//
// label21
//
this.label21.AutoSize = true;
this.label21.Location = new System.Drawing.Point(238, 76);
this.label21.Name = "label21";
this.label21.Size = new System.Drawing.Size(31, 13);
this.label21.TabIndex = 20;
this.label21.Text = "Valor";
//
// txtPedValor
```

```
//
this.txtPedValor.DecimalPlaces = 2;
this.txtPedValor.Location = new System.Drawing.Point(275, 73);
this.txtPedValor.Maximum = new decimal(new int[] {
-1530494976,
232830,
0,
0});
this.txtPedValor.Minimum = new decimal(new int[] {
-727379968,
232,
0,
-2147483648});
this.txtPedValor.Name = "txtPedValor";
this.txtPedValor.Size = new System.Drawing.Size(61, 20);
this.txtPedValor.TabIndex = 19;
//
// txtPedQtd
//
this.txtPedQtd.DecimalPlaces = 2;
this.txtPedQtd.Location = new System.Drawing.Point(166, 73);
this.txtPedQtd.Maximum = new decimal(new int[] {
-1530494976,
232830,
0,
0});
this.txtPedQtd.Minimum = new decimal(new int[] {
-727379968,
232,
0,
-2147483648});
this.txtPedQtd.Name = "txtPedQtd";
this.txtPedQtd.Size = new System.Drawing.Size(61, 20);
this.txtPedQtd.TabIndex = 18;
//
// label20
//
this.label20.AutoSize = true;
this.label20.Location = new System.Drawing.Point(136, 76);
this.label20.Name = "label20";
this.label20.Size = new System.Drawing.Size(24, 13);
this.label20.TabIndex = 17;
this.label20.Text = "Qtd";
//
// label19
//
this.label19.AutoSize = true;
this.label19.Location = new System.Drawing.Point(2, 76);
this.label19.Name = "label19";
this.label19.Size = new System.Drawing.Size(51, 13);
this.label19.TabIndex = 16;
this.label19.Text = "Cód.Prod";
//
// txtPedProd
//
this.txtPedProd.Location = new System.Drawing.Point(59, 73);
```

```
this.txtPedProd.Name = "txtPedProd";
this.txtPedProd.Size = new System.Drawing.Size(71, 20);
this.txtPedProd.TabIndex = 15;
//
// label18
//
this.label18.AutoSize = true;
this.label18.Location = new System.Drawing.Point(3, 61);
this.label18.Name = "label18";
this.label18.Size = new System.Drawing.Size(30, 13);
this.label18.TabIndex = 14;
this.label18.Text = "Itens";
//
// lstPedItens
//
this.lstPedItens.Location = new System.Drawing.Point(4, 100);
this.lstPedItens.Name = "lstPedItens";
this.lstPedItens.Size = new System.Drawing.Size(422, 118);
this.lstPedItens.TabIndex = 13;
this.lstPedItens.UseCompatibleStateImageBehavior = false;
//
// txtPedCli
//
this.txtPedCli.Location = new System.Drawing.Point(60, 34);
this.txtPedCli.Name = "txtPedCli";
this.txtPedCli.Size = new System.Drawing.Size(184, 20);
this.txtPedCli.TabIndex = 12;
//
// label17
//
this.label17.AutoSize = true;
this.label17.Location = new System.Drawing.Point(3, 37);
this.label17.Name = "label17";
this.label17.Size = new System.Drawing.Size(39, 13);
this.label17.TabIndex = 11;
this.label17.Text = "Cliente";
//
// txtPedData
//
this.txtPedData.Location = new System.Drawing.Point(60, 8);
this.txtPedData.Mask = "00/00/0000";
this.txtPedData.Name = "txtPedData";
this.txtPedData.Size = new System.Drawing.Size(71, 20);
this.txtPedData.TabIndex = 10;
this.txtPedData.ValidatingType = typeof(System.DateTime);
//
// label16
//
this.label16.AutoSize = true;
this.label16.Location = new System.Drawing.Point(3, 11);
this.label16.Name = "label16";
this.label16.Size = new System.Drawing.Size(30, 13);
this.label16.TabIndex = 0;
this.label16.Text = "Data";
//
// tabPage5
```

```
//
this.tabPage5.Controls.Add(this.label22);
this.tabPage5.Controls.Add(this.listView1);
this.tabPage5.Controls.Add(this.btnConsPedFiltr);
this.tabPage5.Controls.Add(this.txtConsPedData);
this.tabPage5.Location = new System.Drawing.Point(4, 22);
this.tabPage5.Name = "tabPage5";
this.tabPage5.Size = new System.Drawing.Size(429, 266);
this.tabPage5.TabIndex = 4;
this.tabPage5.Text = "Consulta Pedidos";
this.tabPage5.UseVisualStyleBackColor = true;
//
// label22
//
this.label22.AutoSize = true;
this.label22.Location = new System.Drawing.Point(9, 16);
this.label22.Name = "label22";
this.label22.Size = new System.Drawing.Size(30, 13);
this.label22.TabIndex = 15;
this.label22.Text = "Data";
//
// listView1
//
this.listView1.Location = new System.Drawing.Point(12, 41);
this.listView1.Name = "listView1";
this.listView1.Size = new System.Drawing.Size(403, 213);
this.listView1.TabIndex = 14;
this.listView1.UseCompatibleStateImageBehavior = false;
//
// btnConsPedFiltr
//
this.btnConsPedFiltr.Location = new System.Drawing.Point(335, 9);
this.btnConsPedFiltr.Name = "btnConsPedFiltr";
this.btnConsPedFiltr.Size = new System.Drawing.Size(80, 26);
this.btnConsPedFiltr.TabIndex = 13;
this.btnConsPedFiltr.Text = "Filtrar";
this.btnConsPedFiltr.UseVisualStyleBackColor = true;
//
// txtConsPedData
//
this.txtConsPedData.Location = new System.Drawing.Point(45, 13);
this.txtConsPedData.Mask = "00/00/0000";
this.txtConsPedData.Name = "txtConsPedData";
this.txtConsPedData.Size = new System.Drawing.Size(71, 20);
this.txtConsPedData.TabIndex = 12;
this.txtConsPedData.ValidatingType = typeof(System.DateTime);
//
// tabPage6
//
this.tabPage6.Controls.Add(this.btnConsCliFiltr);
this.tabPage6.Controls.Add(this.listView2);
this.tabPage6.Location = new System.Drawing.Point(4, 22);
this.tabPage6.Name = "tabPage6";
this.tabPage6.Size = new System.Drawing.Size(429, 266);
this.tabPage6.TabIndex = 5;
this.tabPage6.Text = "Consulta Clientes";
```

```
this.tabPage6.UseVisualStyleBackColor = true;
//
// btnConsCliFiltr
//
this.btnConsCliFiltr.Location = new System.Drawing.Point(337, 7);
this.btnConsCliFiltr.Name = "btnConsCliFiltr";
this.btnConsCliFiltr.Size = new System.Drawing.Size(80, 26);
this.btnConsCliFiltr.TabIndex = 14;
this.btnConsCliFiltr.Text = "Filtrar";
this.btnConsCliFiltr.UseVisualStyleBackColor = true;
//
// listView2
//
this.listView2.Location = new System.Drawing.Point(8, 39);
this.listView2.Name = "listView2";
this.listView2.Size = new System.Drawing.Size(409, 215);
this.listView2.TabIndex = 0;
this.listView2.UseCompatibleStateImageBehavior = false;
//
// groupBox2
//
this.groupBox2.Controls.Add(this.lstNodos);
this.groupBox2.Location = new System.Drawing.Point(455, 12);
this.groupBox2.Name = "groupBox2";
this.groupBox2.Size = new System.Drawing.Size(209, 391);
this.groupBox2.TabIndex = 2;
this.groupBox2.TabStop = false;
this.groupBox2.Text = "Nodos Conhecidos";
//
// lstNodos
//
this.lstNodos.FormattingEnabled = true;
this.lstNodos.Location = new System.Drawing.Point(10, 21);
this.lstNodos.Name = "lstNodos";
this.lstNodos.Size = new System.Drawing.Size(198, 355);
this.lstNodos.TabIndex = 0;
//
// Form1
//
this.AutoScaleDimensions = new System.Drawing.SizeF(6F, 13F);
this.AutoScaleMode = System.Windows.Forms.AutoScaleMode.Font;
this.ClientSize = new System.Drawing.Size(676, 451);
this.Controls.Add(this.groupBox2);
this.Controls.Add(this.tabControl1);
this.Controls.Add(this.groupBox1);
this.Name = "Form1";
this.Text = "Exemplo Cliente";
this.Load += new System.EventHandler(this.Form1_Load);
this.groupBox1.ResumeLayout(false);
this.groupBox1.PerformLayout();
this.tabControl1.ResumeLayout(false);
this.tabPage1.ResumeLayout(false);
this.tabPage1.PerformLayout();
this.tabPage2.ResumeLayout(false);
this.tabPage2.PerformLayout();
this.tabPage3.ResumeLayout(false);
```

```
    this.tabPage3.PerformLayout();
    ((System.ComponentModel.ISupportInitialize)(this.txtEstqQtd)).EndInit();
    this.tabPage4.ResumeLayout(false);
    this.tabPage4.PerformLayout();
    ((System.ComponentModel.ISupportInitialize)(this.txtPedValor)).EndInit();
    ((System.ComponentModel.ISupportInitialize)(this.txtPedQtd)).EndInit();
    this.tabPage5.ResumeLayout(false);
    this.tabPage5.PerformLayout();
    this.tabPage6.ResumeLayout(false);
    this.groupBox2.ResumeLayout(false);
    this.ResumeLayout(false);

}

#endregion

private System.Windows.Forms.GroupBox groupBox1;
private System.Windows.Forms.TextBox txtPorta;
private System.Windows.Forms.TextBox txtEndereco;
private System.Windows.Forms.Button btnAdicionar;
private System.Windows.Forms.Button btnTestar;
private System.Windows.Forms.Label label2;
private System.Windows.Forms.Label label1;
private System.Windows.Forms.TabControl tabControl1;
private System.Windows.Forms.TabPage tabPage1;
private System.Windows.Forms.TabPage tabPage2;
private System.Windows.Forms.GroupBox groupBox2;
private System.Windows.Forms.ListBox lstNodos;
private System.Windows.Forms.Button btnProdAdd;
private System.Windows.Forms.Label label8;
private System.Windows.Forms.TextBox txtProdObs;
private System.Windows.Forms.TextBox txtProdPeso;
private System.Windows.Forms.Label label7;
private System.Windows.Forms.Label label6;
private System.Windows.Forms.Label label5;
private System.Windows.Forms.TextBox txtProdUnid;
private System.Windows.Forms.TextBox txtProdValor;
private System.Windows.Forms.Label label4;
private System.Windows.Forms.TextBox txtProdNome;
private System.Windows.Forms.TextBox txtProdCod;
private System.Windows.Forms.Label label3;
private System.Windows.Forms.Label label12;
private System.Windows.Forms.TextBox txtCliSenha;
private System.Windows.Forms.Label label11;
private System.Windows.Forms.TextBox txtCliEmail;
private System.Windows.Forms.Label label10;
private System.Windows.Forms.Label label9;
private System.Windows.Forms.TextBox txtCliNome;
private System.Windows.Forms.TextBox txtCliCodNac;
private System.Windows.Forms.TabPage tabPage3;
private System.Windows.Forms.TabPage tabPage4;
private System.Windows.Forms.TabPage tabPage5;
private System.Windows.Forms.Button btnCliAdd;
private System.Windows.Forms.MaskedTextBox txtCliNasc;
private System.Windows.Forms.Label label13;
private System.Windows.Forms.Button btnEstqSet;
```

```
        private System.Windows.Forms.Button btnEstqInc;
        private System.Windows.Forms.NumericUpDown txtEstqQtd;
        private System.Windows.Forms.Label label15;
        private System.Windows.Forms.TextBox txtEstqCod;
        private System.Windows.Forms.Label label14;
        private System.Windows.Forms.Label label19;
        private System.Windows.Forms.TextBox txtPedProd;
        private System.Windows.Forms.Label label18;
        private System.Windows.Forms.ListView lstPedItens;
        private System.Windows.Forms.TextBox txtPedCli;
        private System.Windows.Forms.Label label17;
        private System.Windows.Forms.MaskedTextBox txtPedData;
        private System.Windows.Forms.Label label16;
        private System.Windows.Forms.Button btnPedAdicionar;
        private System.Windows.Forms.Button btnPedAdd;
        private System.Windows.Forms.Label label21;
        private System.Windows.Forms.NumericUpDown txtPedValor;
        private System.Windows.Forms.NumericUpDown txtPedQtd;
        private System.Windows.Forms.Label label20;
        private System.Windows.Forms.ListView listView1;
        private System.Windows.Forms.Button btnConsPedFiltr;
        private System.Windows.Forms.MaskedTextBox txtConsPedData;
        private System.Windows.Forms.TabPage tabPage6;
        private System.Windows.Forms.ListView listView2;
        private System.Windows.Forms.Label label22;
        private System.Windows.Forms.Button btnConsCliFiltr;
    }
}
```

frmPrincipal.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using Barsa;

namespace ExemploCliente
{
    public partial class frmPrincipal : Form
    {
        private List<Nodo> lNodos = new List<Nodo>();
        private DealerSR.DealerServiceSoapClient w;

        public frmPrincipal()
        {
            InitializeComponent();
            w = new DealerSR.DealerServiceSoapClient();
        }
```

```
private void btnTestar_Click(object sender, EventArgs e)
{
   //DealerSR.DealerServiceSoapClient w = new DealerSR.DealerServiceSoapClient();
   //DealerSR.DealerService = new DealerSR.DealerService();
   //DealerSR.u
   MessageBox.Show(w.TesteBD());

}

private void btnAdicionar_Click(object sender, EventArgs e)
{
   Nodo n = new Nodo();
   n.Endereco = txtEndereco.Text;
   n.Porta = txtPorta.Text;
   lNodos.Add(n);
}

private void Form1_Load(object sender, EventArgs e)
{

}

private void CarregarNodos() {
   lstNodos.Items.Clear();

   foreach(Nodo nd in lNodos){
      lstNodos.Items.Add(nd.Endereco + ":" + nd.Porta);
   }
}

private void btnPedAdicionar_Click(object sender, EventArgs e)
{
   Pedido p = new Pedido();
   p.Data = DateTime.Parse(txtPedData.Text);
   p.Itens = new List<Item>();
   p.Nodo = lNodos.ElementAt(0);
   //TODO
}

private void btnProdAdd_Click(object sender, EventArgs e)
{
   Produto p = new Produto();
   p.Codigo = txtProdCod.Text;
   p.Nome = txtProdNome.Text;
   p.Observacoes = txtProdObs.Text + "- WINDOWS";
   p.ID = null;
   p.Peso = double.Parse(txtProdPeso.Text);
   p.Preco = double.Parse(txtProdValor.Text);
   p.Unidade = txtProdUnid.Text;

   //DealerSR.DealerServiceSoapClient w = new DealerSR.DealerServiceSoapClient();
   w.AdicionarProduto(Funcoes.Serializar(p));

   MessageBox.Show("Produto Cadastrado!");
```

```
            limparProduto();
        }

        private void limparProduto() {
            txtProdCod.Text = "";
            txtProdNome.Text = "";
            txtProdObs.Text = "";
            txtProdPeso.Text = "0";
            txtProdUnid.Text = "";
            txtProdValor.Text = "0,00";
        }

        private void limparCliente() {
            txtCliCodNac.Text = "";
            txtCliEmail.Text = "";
            txtCliNasc.Text = "";
            txtCliNome.Text = "";
            txtCliSenha.Text = "";
        }

        private void btnCliAdd_Click(object sender, EventArgs e)
        {
            Cliente c = new Cliente();
            c.Bloqueado = false;
            c.CodigoDeAcesso = txtCliSenha.Text;
            c.CodigoNacional = txtCliCodNac.Text;
            c.DataNascimento = Convert.ToDateTime (txtCliNasc.Text);
            c.EMail = txtCliEmail.Text;
            c.ID = null;
            c.MensagemBloqueio = "";
            c.Nome = txtCliNome.Text;

            w.AdicionarCliente(Funcoes.Serializar(c));


            MessageBox.Show("Cliente Cadastrado!");
            limparCliente();
        }


    }
}
```

Cliente Android

Webserv.js

```
        var servidor;
        var porta;
        var servico;
        var resultadoWS;
```

```
	function inicializarWS(_server, _porta){
		//alert('inicial');
		servidor = _server;
		porta = _porta;

		servico = 'http://' + servidor;
		if(porta.length > 0){
			servico = servico + ':' + porta;
		}

		servico = servico + '/rf2server.asmx';
		partes = new Array();
	}


	function testeWS(){
		//alert('Teste Funcao');
		jQuery.support.cors = true;
		// http://187.4.227.90/rf2server.asmx
		var webMethod = servico + '?op=HelloWorld';
		//alert(webMethod);
		var soapMessage = '<?xml version="1.0" encoding="utf-8"?><soap:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"> <soap:Body> <HelloWorld
xmlns="http://tempuri.org/" /> </soap:Body></soap:Envelope>'
		$.ajax({
	   type: "POST",
	   url: webMethod,
	   contentType: 'text/xml; charset=ansi',
	   dataType: "xml",
	   data: soapMessage,
	   success: function (data, status, req) {
	      //alert("OK " + req.responseText);
	   },
	   error: function (XMLHttpRequest, textStatus, errorThrown) {
	      alert("Unavailable " + errorThrown);
	   }
	});

	//alert('3');
	 }


	 function processarRetorno(funcao, xml){
		//alert('processarRetorno' + xml);
	 var resultado ='';
	 var ix = xml.indexOf('<' + funcao + 'Result>');
	 var ix2 = xml.indexOf('</' + funcao + 'Result>');
	 //alert(ix + ', ' + ix2);
	 if (ix > -1 && ix2 > -1){
		var l = ('<' + funcao + 'Result>').length;
		ix = ix + l;
		resultado = xml.substring(ix, ix2);
	 }
	 //alert(resultado);
```

```
	return resultado;
	}

	function teste2(){
		var strChave = processarRetorno('PegaChaveRemessaPCPocket', resultadoWS);
		//alert(strChave);
		tmpParte = 1;
		pegarPartesImportacao(strChave, 1);

		//var webMethod = servico + '?op=HelloWorld';
		//alert(webMethod);
		//var soapMessage = '<?xml version="1.0" encoding="utf-8"?><soap:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"> <soap:Body> <HelloWorld
xmlns="http://tempuri.org/" /> </soap:Body></soap:Envelope>'
		//chamarService(webMethod, soapMessage);
	}

	function chamarService(webMethod, soapMessage){
		//alert(webMethod);
		$.ajax({
	  type: "POST",
	  url: webMethod,
	  contentType: 'text/xml; charset=ansi',
	  dataType: "xml",
	  data: soapMessage,
	  success: function (data, status, req) {
	     //alert("OK " + req.responseText);
	     resultadoWS = req.responseText;
	     //$("#btnSucesso").click();
	     finalizouWS();
	  },
	  error: function (XMLHttpRequest, textStatus, errorThrown) {
		if(webMethod.indexOf("PegaEstoquesProdutos") == -1){
			alert("ERRO: Serviço Indisponível " + errorThrown);
		}

	  }
	});

	//alert('FIM SERVICE');
	}

	function pegarResultadoService(){
		return resultadoWS;
	}

	function exportarPedido(xmlExp){

		var webMethod = servico + '?op=FazerPedido;

		var soapMessage = '<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"> <soap:Body> <FazerPedido
```

```
xmlns="http://tempuri.org/"> <XMLPedido>' + xmlExp + '</XMLPedido> </FazerPedido> </soap:Body>
</soap:Envelope>';

                    chamarService(webMethod, soapMessage);
          }


Function gerarXMLPedido(p){
      var xmlExp = ‘’;
      xmlExp = xmlExp + '<Pedido>';
      xmlExp = xmlExp + '<Vendedor>';
      xmlExp = xmlExp + p.getVendedor()
      xmlExp = xmlExp + '</Vendedor>';
      xmlExp = xmlExp + '<Cliente>';
      xmlExp = xmlExp + p.getCliente();
      xmlExp = xmlExp + '</Cliente>';
      xmlExp = xmlExp + '<Itens>';
      itns = peds[i].getItens();
      for (j = 0; j < itns.length; j++) {
         xmlExp = xmlExp + '<Item>';
         xmlExp = xmlExp + '<Quantidade>'+ itns[j].getQtd() + '</Quantidade>';
         xmlExp = xmlExp + '<Produto>';
         xmlExp = xmlExp + itns[j].getCodigo();
         xmlExp = xmlExp + '</Produto>';
         xmlExp = xmlExp + '</Item>';
      }
      xmlExp = xmlExp + '</Itens>';
      xmlExp = xmlExp + '<Data>' + p.getData() + 'T00:00:00</Data>';
      xmlExp = xmlExp + '<Codigo>'  + p.getCodigo() +'</Codigo>';
      xmlExp = xmlExp + '</Pedido>';

}
```

Index.html

```
<!DOCTYPE HTML>
<html>
<head>
   <title></title>
<script type="text/javascript" charset="utf-8" src="cordova-2.1.0.js"></script>
<script type="text/javascript" charset="utf-8" src="jquery.mobile-1.0.1.min.js"></script>
<script type="text/javascript" charset="utf-8" src="jquery-1.6.4.js"></script>
<script language="javascript" src="codigo/webServ.js"></script>
   <style type="text/css">
        body
        {
            font-family: Arial, Helvetica, sans-serif;
         }

        .Titulo
        {
          background-color: #C0C0C0;
                font-size: x-large;
         }
```

```
    </style>
    <script type="text/javascript" language="javascript">
        var produtos;

function Pedido() {
    var vendedor;
    var cliente;
    var itens;
    var data;
    var codigo;
    var totalProds;
    var total;

    this.getVendedor = getVendedor;
    this.getCliente = getCliente;
    this.getItens = getItens;
    this.getData = getData;
    this.getCodigo = getCodigo;
    this.getTotalProds = getTotalProds;
    this.getTotal = getTotal;

    this.setVendedor = setVendedor;
    this.setCliente = setCliente;
    this.setItens = setItens;
    this.setData = setData;
    this.setCodigo = setCodigo;
    this.setTotalProds = setTotalProds;
   this.setTotal = setTotal;


    function getVendedor() {
        return vendedor;
    }

    function getCliente() {
        return cliente;
    }

    function getItens() {
        return itens;
    }

    function getData() {
        return data;
    }

    function getCodigo() {
        return codigo;
    }

    function getTotalProds() {
        return totalProds;
    }

    function getTotal() {
        return total;
```

```
    }


    function setVendedor(_vendedor) {
        vendedor = _vendedor;
    }

    function setCliente(_cliente) {
        cliente = _cliente;
    }

    function setItens(_itens) {
        itens = _itens;
    }

    function setData(_data) {
        data = _data;
    }

    function setCodigo(_codigo) {
        codigo = _codigo;
    }


    function setTotalProds(_totalProds) {
        totalProds = _totalProds;
    }

    function setTotal(_total) {
        total = _total;
    }

}

function Item(){
            var codigo;
            var qtd;

            this.getCodigo = getCodigo;
            this.getQtd = getQtd;
            this.setCodigo = setCodigo;
            this.setQtd = setQtd;


    function getCodigo() {
      return codigo;
    }

    function getQtd() {
      return qtd;
    }



    function setCodigo(_codigo) {
      codigo = _codigo;
```

```
    }

    function setQtd(_qtd) {
      qtd = _qtd;
    }

}

        function gravarPedido() {
            meuPedido = new Pedido();
            meuPedido.setVendedor($("#txtVendedor").val());
            var d=new Date();
            var dtAtual = d.getFullYear() + '-' + formatarNumero((d.getMonth() + 1)) + '-' +
            formatarNumero(d.getDate());
            meuPedido.setData(dtAtual);
            meuPedido.setItens(itens);
            meuPedido.setCliente(($("#txtCliente").val()););
            nrPedido = parseInt(nrPedido) + 1;
            meuPedido.setCodigo(nrPedido);
            var xml = gerarXMLPedido(meuPedido);
            exportarPedido(xml);
            alert('Pedido Cadastrado');
            limpar();

        }

        function adicionarProduto() {
            itm = new Item();
            itm.setCodigo($("#txtProduto ").val());
            itm.setQtd($("#txtQtd ").val());
            $("#txtItens").val($("#txtItens").val('') + '/n' + itm.getQtd() + ' X ' + itm.getCodigo());

            itm [itm.length] = itm;

        }

        function limpar() {
            itens = new Array();
            $("#txtCliente").val('');
            $("#txtVendedor").val('');
            $("#txtItens").val('');
            limparDadosProduto();
        }

        function limparDadosProduto() {
            $("#txtQtd").val('');
            $("#txtProduto").val('');

        }
    </script>
</head>
<body>
    <table width="100%">
        <tr>
            <td class="Titulo" colspan="2">
                PEDIDO
```

```
            </td>
        </tr>
        <tr>
            <td>
                 
            </td>
        </tr>
        <tr>
            <td>
                Cliente
            </td>
            <td>
                <input id="txtCliente" style="width:100%" type="text" />
            </td>
        </tr>
         <tr>
            <td>
                Vendedor
            </td>
            <td>
                <input id="txtVendedor" style="width:100%" type="text" />
            </td>
        </tr>
         <tr>
            <td>
                Produto
            </td>
            <td>
                <input id="txtProduto" style="width:100%" type="text" />
            </td>
        </tr>
        <tr>
            <td colspan="2">
                <table width="100%" cellpadding="0" cellspacing="0">
                    <tr>
                        <td>
                            Qtd
                        </td>
                        <td>
                            <input id="txtQtd" style="width:100%" type="text" />
                        </td>
                        <td align="right">
                            <input id="btnAddProduto" type="button" value="Adicionar"
onclick="adicionarProduto();" />
                        </td>
                    </tr>
                </table>
            </td>
        </tr>
         <tr>
            <td colspan="2">
                <textarea id="txtItens" cols="50" rows="50" ></textarea>
            </td>
        </tr>
          <tr>
            <td>
```

```
            Web Service
          </td>
          <td>
            <input id="txtWebService" style="width:100%" type="text" />
          </td>
        </tr>
         <tr>
          <td colspan="2" align="right">
            <input id="btnGravar" type="button" value="Gravar" onclick="gravarPedido();" />
            <input id="btnLimpar" type="button" value="Limpar" onclick="limpar();" />
          </td>
        </tr>
    </table>
</body>
</html>
```

APÊNDICE B – Artigo

Sistema de alta disponibilidade para gerenciamento de pedidos e estoque

Mariell Schappo

Depto. de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Caixa Postal 476 – 88.040-900 – Florianópolis – SC – Brasil.

mariellschappo@gmail.com

R**esumo.** Este artigo refere-se ao trabalho de conclusão de curso, que consiste no projeto e implementação de um sistema de geração de pedidos compatível com sistemas ERP mais comuns através da comunicação via web services. *Na maioria dos sistemas de venda e controle gerencial ocorre um grande número de operações relacionadas à efetivação de pedidos, enquanto outras áreas do sistema têm um fluxo moderado, como emissão de notas fiscais e relatórios gerenciais. Através de um sistema especializado, o grande fluxo de pedidos será gerenciado através de vários servidores que se comunicam para manter seus dados consistentes, sem perder a autonomia individual para realizar o pedido. Para cumprir estas especificações, cada servidor contará com um estoque local e um estoque global, e um servidor poderá efetuar instantaneamente um pedido se possuir estoque local. Caso contrário, terá que solicitar uma transferência de outro servidor. A comunicação entre os servidores se dá via troca de mensagens.*

Abstract. *This paper refers to the course conclusion project, which consists of the design and implementation of a system for generating applications compatible with the most common ERP systems through communication via web services. In most sales systems and management control there is a large number of operations related to the execution of orders, while other areas of the system have a moderate flow, such as issuing invoices and management reports. Through a specialized system, the large flow of requests will be managed through several servers that communicate to keep your data consistent, without losing individual autonomy to perform the request. To accomplish these specifications, each server will have a local inventory and a global inventory, and a server can instantly place an order if you have local stock. Otherwise, you will have to request a transfer from another server. Communication between servers is via messaging.*

1. Introdução

Com o passar do tempo o setor de comércio está se modificando, concentrando-se cada vez mais em grandes empresas. Normalmente uma empresa, mesmo sedimentada, concentra seus dados em um único sistema de grande porte e geralmente em um único servidor com grande capacidade e com backups espalhados geograficamente.

O problema deste tipo desse modelo clássico é que mesmo com grande poder computacional do servidor, os clientes necessitam de uma boa estrutura de comunicação em rede para que o sistema tenha um bom desempenho para seus usuários finais, o que nem sempre é viável.

O objetivo deste projeto de conclusão de curso é desenvolver um módulo de pedidos em um ambiente distribuído que possa ser facilmente incrementado à medida que a demanda do sistema cresça, ou seja, o sistema deve balancear as solicitações de clientes através do uso de vários servidores de modo que todos eles forneçam dados precisos em um tempo rápido, mesmo quando o sistema encontra-se muito distribuído geograficamente.

2. Utilização de web services

Para a comunicação do sistema com o mundo externo é utilizado web service, utilizando o protocolo SOAP. Mensagens SOAP são utilizadas principalmente baseadas em HTTP, porém o protocolo pode ser também em conjunto com outros protocolos de transporte. De modo geral, este protocolo determina de que forma o XML deve ser estruturado para que transporte o conteúdo das mensagens com um padrão reconhecido pelo destinatário independente da linguagem em que o software foi construído. A figura 1 mostra o funcionamento do web service.

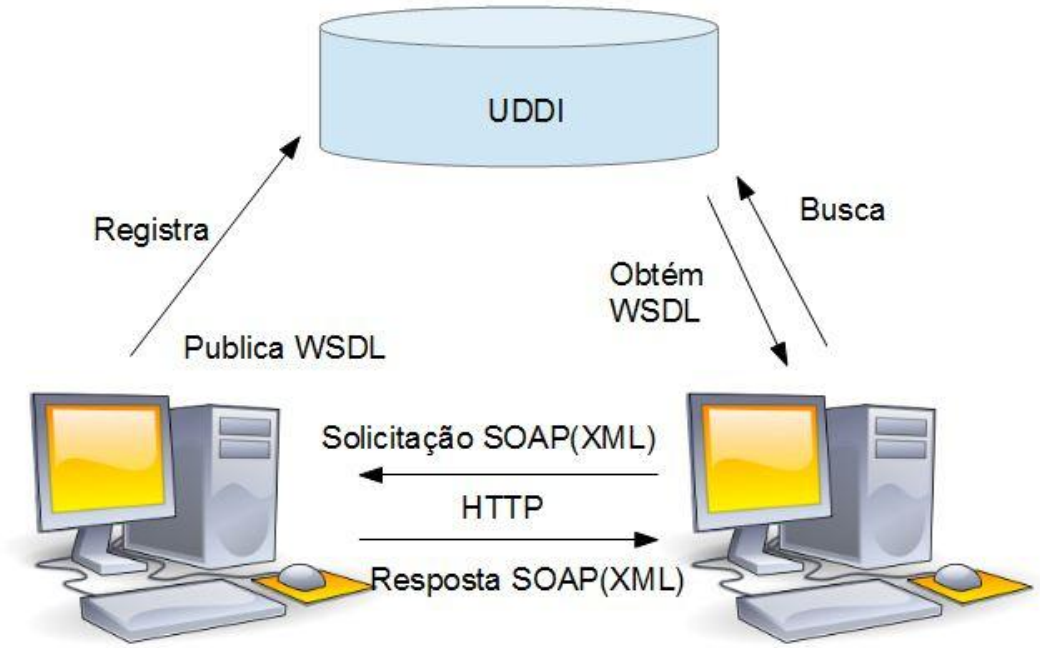

Figura 1. ***Esquema do funcionamento do web service***

2. Visão geral do sistema

O sistema desenvolvido neste projeto provê a criação de pedidos de compra através de uma estrutura combinada de vários nodos, em contrapartida ao modelo clássico cliente-servidor. O sistema permite comunicação do cliente e do ERP com qualquer nodo pertencente ao conjunto de nodos operantes. O ERP alimenta o sistema com dados básicos como cadastro de produtos e quantidade de estoque, além de consultar os pedidos realizados pelo sistema. O cliente, por sua vez, é quem vai realizar os pedidos, cadastrar novos clientes e consultar preços.

A figura 2 demonstra como as partes interagem entre si, formando o sistema de geração de pedidos e controle de estoque, em um ambiente distribuído geograficamente.

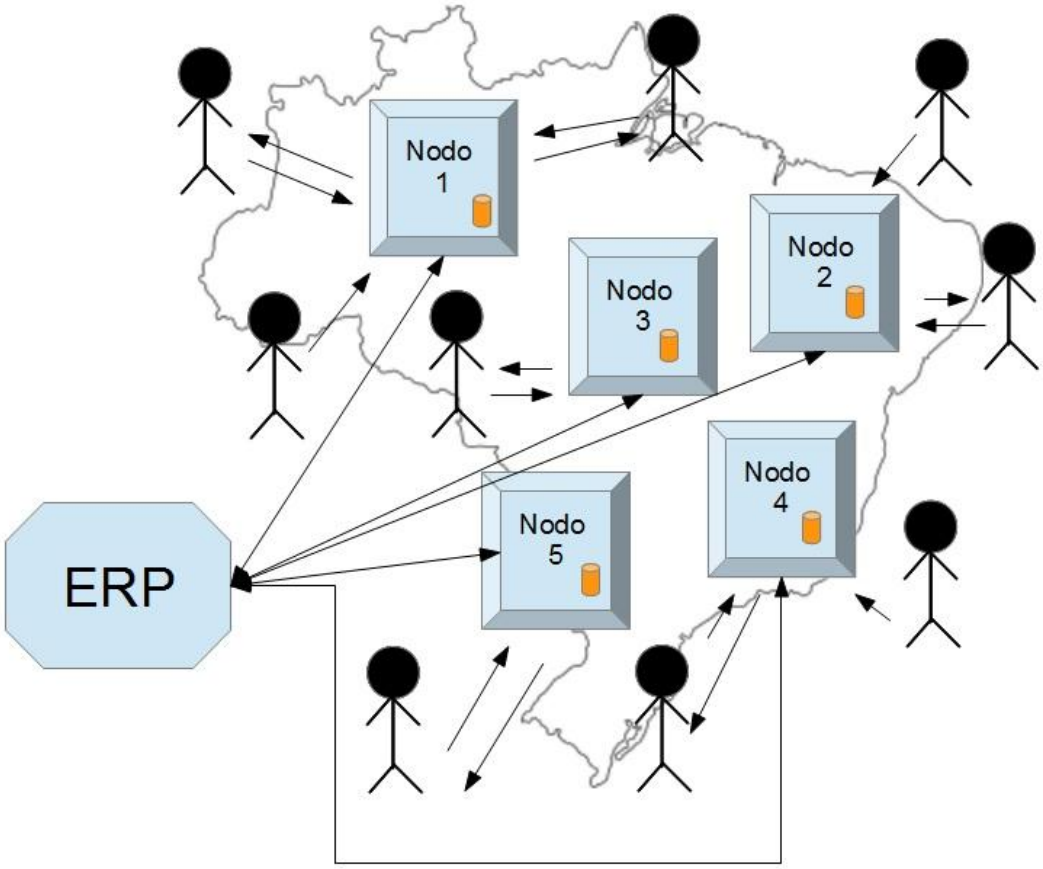

Figura 2. ***Esquema do sistema***

3. Sistemática para o controle de estoque

Como já explicado, a função deste sistema é a criação de pedidos de maneira distribuída. Dois itens devem ser necessariamente contemplados para que o sistema possa ter sua utilização justificada: a velocidade das consultas e gravações de um pedido em ambiente com restrições de comunicação deve ser maior em relação a um modelo cliente-servidor simples; e o sistema não pode, de nenhuma maneira, vender um item que não consta no estoque.

Para que haja um controle de estoque, em um sistema que roda paralelamente em vários nodos, foi adotada uma estratégia que visa resolver este problema sem causar um grande *overhead* no sistema, demonstrada na Figura 3.

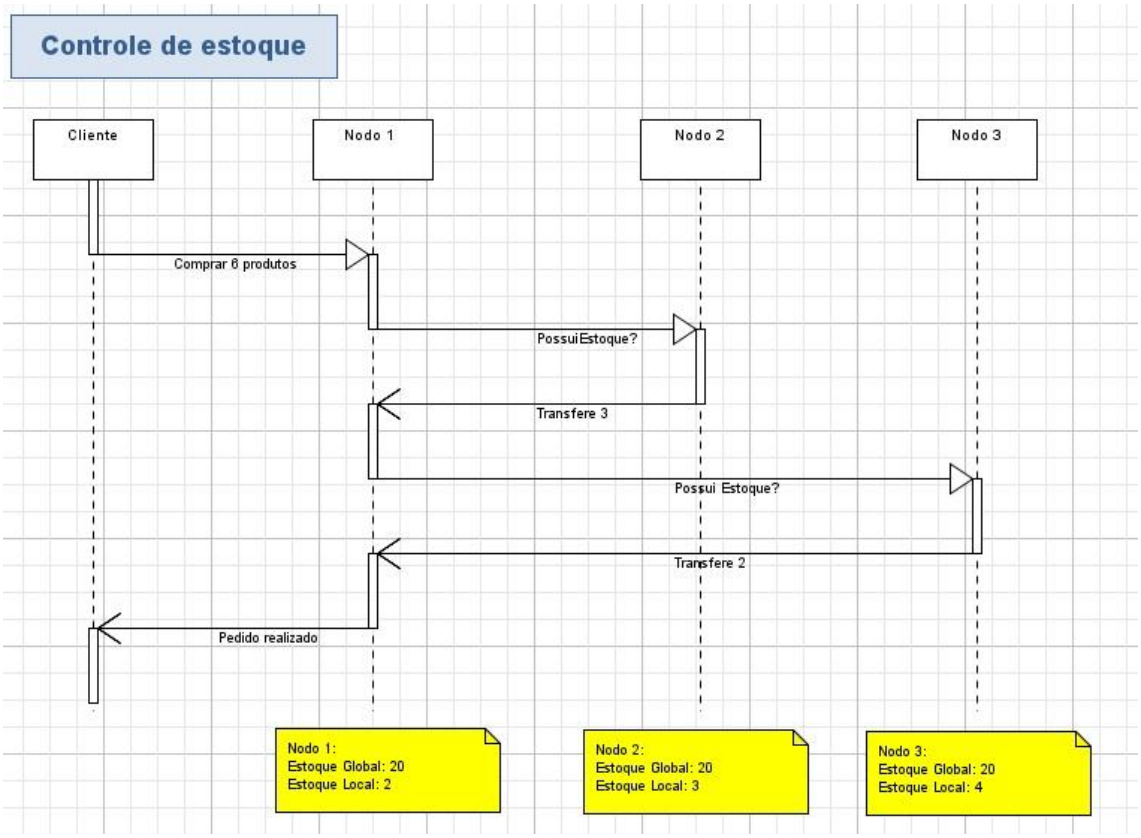

Figura 3. ***Diagrama de sequência do controle de estoque***

4. Conclusão

A solução implementada neste trabalho mostra como é possível dividir as funcionalidades, de forma a atender melhor as partes mais críticas de um sistema de gerenciamento através de um sistema especialista e facilmente integrável com outros sistemas. Apesar de ser uma solução direcionada a um caso bem específico, pode servir como base para vários problemas comuns encontrado em empresas grandes, principalmente em países onde a infraestrutura tecnológica ainda representa um entrave no desenvolvimento empresarial. No decorrer deste projeto percebe-se o quão importante é utilizar padrões bem conhecidos em diversas tecnologias, para limitar o mínimo possível os requisitos tecnológicos exigidos no funcionamento do sistema. A forma como foi elaborado, faz com que este sistema seja uma caixa preta com uma comunicação externa padronizada (Web Services), o que possibilita uma integração facilitada com grande parte das plataformas existentes, facilitando a absorção deste sistema pelo mercado.

Referências

1. COULOURIS, George; DOLLIMORE, Jean; KINDBERG, Tim. **Sistemas Distribuídos Conceitos e Projetos**, quarta edição, Editora Bookman.
2. BECKER, CLARO e SOBRAL. **Web Services e XML , Um Novo Paradigma da Computação Distribuída**.Disponível em http://homes.dcc.ufba.br/~dclaro/download/ArtigoWebServices.pdf

3. BERGAMASCHI, Sidnei. **Um estudo sobre projetos de implementação de sistemas para gestão empresarial**. 1999. Dissertação (Mestrado em Métodos Quantitativos) - Faculdade de Economia, Administração e Contabilidade, Universidade de São Paulo, São Paulo, 1999. Disponível em: <http://www.teses.usp.br/teses/disponiveis/12/12133/tde-27122003-224740/>. Acesso em: 2012-07-03.