UNIVERSIDADE FEDERAL DE SANTA CATARINA

COMPUTAÇÃO DISTRIBUÍDA DE ALTO DESEMPENHO - UM ESTUDO DE CASO DE ACESSO MÓVEL

Antônio Carlos Venâncio Júnior
Rodrigo Grumiche Silva

UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

COMPUTAÇÃO DISTRIBUÍDA DE ALTO DESEMPENHO - UM ESTUDO DE CASO DE ACESSO MÓVEL

Autores:
Antônio Carlos Venâncio Júnior
Rodrigo Grumiche Silva

Orientador:
Prof. Mário Antônio Ribeiro Dantas, PhD

Banca Examinadora:
Prof. Dr. José Eduardo De Lucca
Prof. Dr. Vitório Bruno Mazzola

Palavras-chave:
Web Services, Computação Móvel, Clusters

Florianópols, 13 de fevereiro de 2004

UNIVERSIDADE FEDERAL DE SANTA CATARINA
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE CIÊNCIAS DA COMPUTAÇÃO

COMPUTAÇÃO DISTRIBUÍDA DE ALTO DESEMPENHO - UM ESTUDO DE CASO DE ACESSO MÓVEL

Este trabalho de conclusão de curso foi julgado adequado à obtenção do grau de Bacharelado em Ciências da Computação e aprovado em sua forma final pelo Curso de Ciências da Computação da Universidade Federal de Santa Catarina.

Florianópolis – SC, 13 de fevereiro de 2004.

_____
Prof. Mário Antônio Ribeiro Dantas, PhD
Universidade Federal de Santa Catarina

_____
Prof. Dr. José Eduardo De Lucca
Universidade Federal de Santa Catarina

_____
Prof. Dr. Vitório Bruno Mazzola
Universidade Federal de Santa Catarina

DEDICATÓRIA

Antônio

Dedico este trabalho a todos aqueles que acreditaram em mim e aos que contribuem de alguma forma para o desenvolvimento de softwares de código livre e de qualidade.

Rodrigo

Aos meus pais:
Herminio Manoel Silva Junior
Sandra Maria Silva
Vocês tornaram possível...

AGRADECIMENTOS

Antônio

Agradeço a meus familiares e amigos pelo apoio e compreensão e a todos aqueles que me ajudaram a chegar até aqui (sem esquecer da "tiazinha do RU").

Rodrigo

Aos meus pais e toda a família pelo apoio e consideração que tiveram com minha luta para conseguir realizar este trabalho, e finalmente terminar minha graduação em Ciências da Computação.

Aos amigos pelo incentivo e carinho nos momentos difícies.

A Alexandra por sua compreensão com a minha ausência e por seu amor.

A todos aqueles que diretamente ou indiretamente ajudaram na realização deste trabalho.

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

ABI - Allied Business Inteligence Inc
AMD – Advanced Micro Devices
AMPS - Advanced Mobile Phone Service
ANATEL – Agência Nacional de Telecomunicações
API - Application Program Interface
B2B – Business To Business
BSIG - Bluetooth Special Interest Group
CDMA - Code Division Multiple Access
CRM – Customer Relationship Management
DAL – Device Abastraction Layer
DNS – Domain Name Server
DOM - Document Object Model
DTD - Document Type Definition
E/S – Entrada/Saída
EDGE - Enhanced Data Rates for Global Evolution
EDI - Eletronic Data Interchange
ESS - Extended Service Set
GPL – General Public License
GPRS - General Packet Data Radio Service
GSM - Global System for Mobile Telecommunications
HAL – Hardware Abastraction Layer
HSCSD - High Speed Circuit-Switched Data
HTML - HyperText Markup Language
IBSS - Independent Basic Service Set
IDE - Intelligent Drive Electronics (ou Integrated Drive Electronics)
IEEE - Institute of ElectricalandElectronics Engineers
IP – Internet Protocol

J2ME – Java 2 Platform, Micro Edition
LAM - Local Area Multicomputer
LCD – Liquid Crystal Display
LUI - Linux Utility for cluster Installation
MIMD - Multiple Instruction Multiple Data
MMS – Multimedia Messaging Services
MPI - Message Passing Interface
NFS - Network File System
ODBC - Open Database Connectivity
OSCAR - Open Source Cluster Application Resources
PACE - Palm Application Compatibility Enviroment
PBS - Portable Batch System
PC – Personal Computer
PCI - Peripheral Component Interconnect
PDA - Personal Digital Assistant
PDF - Portable Document Format
PEAR – PHP Extension and Application Repository
PHP – Hipertext Preprocessor
PVM - Parallel Virtual Machine
RPC - Remote Procedure Call
RTF – Rich Text Format
SAX - Simple API fo XML
SCSI - Small Computer System Interface
SGML - Standard Generalized Markup Language
SMS – Short Message System
SOA - Service-Oriented Architecture
SOAP - Simple Object Access Protocol
TCP – Transmission Control Protocol
TDMA - Time Division Multiple Access
TTL – Time To Live
UDDI - Universal Description, Discovery and Integration
UIAS - User Interface Application Shell
W-CDMA - Wideband CDMA
W3C - World Wide Web Consortium
WML - Wireless Markup Language

WSDL - Web Services Description Language

XML - Extensible Markup Language

XSLT - Extensible Style Language Transformation

RESUMO

Este trabalho apresenta um estudo de caso, o qual aborda uma possibilidade de implementação para a disponibilização de um serviço a ser utilizado por vendedores de distribuidoras de produtos. O sistema baseia-se na utilização de Web Services como solução para a disponibilização do serviço, o qual apresenta como partes: um sistema cliente, que foi desenvolvido em C e que pode ser executado a partir de dispositivos móveis utilizando o sistema operacional PalmOS; e um sistema servidor *(*Web Service*),* o qual é executado sobre o servidor web Apache, em conjunto com um banco de dados MySQL, utilizando-se da linguagem de programação PHP e a biblioteca PEAR. Faz-se uma breve revisão bibliográfica sobre as tecnologias envolvidas. Por fim, explana-se as vantagens de se utilizar tal solução.

Palavras-chave: Web Services, computação móvel, clusters

ABSTRACT

This work present a case study about one possible implementation for a service to be used by sellers of goods distributors. The system is based in Web Services as a solution to make the service avaiable for use. The service has two systems: a client, developed in C, which can be executed in any mobile device running PalmOS as operational sistem; and a server (Web Service) that runs over an Apache web server, using the MySQL database to store the data. The server was developed in PHP using the PEAR library. It's made an early bibliography revision about the tecnologies involved. At final, the benefits of this solution are explained.

Keywords: Web Service, mobile computing, clusters

1 INTRODUÇÃO

1.1 Cenário

Chegamos a um nível de desenvolvimento tecnológico onde pode-se facilmente compartilhar recursos computacionais dos mais diversos tipos, e sendo acessíveis por diferentes tipos de equipamentos. As redes de computadores, conjunto de dispositivos, enlaces de comunicação e pacotes de software que permitem as pessoas e equipamentos possam trocar informações [1], representaram uma grande revolução na maneira de como a tecnologia pode ser utilizada para a comunicação e processamento de dados.

Através da grande rede mundial de computadores, a Internet, os usuários acessam serviços fornecidos por organizações ou por outros, a partir de seu computador pessoal, independente de suas localizações. Pode-se verificar o preço das ações de uma empresa na bolsa de valores Nasdaq [2], traduzir um texto em Inglês para Português no Babelfish [3] e muitos outros serviços a partir de suas residências e locais de trabalho. Empresas utilizam a rede para realizar seus processos internos, se relacionar com seus fornecedores (B2B) e com seus clientes (CRM).

Nesse universo, surgiu uma nova tecnologia para disponibilização e acesso a serviços via Internet: *Web Services*. Um *Web Service* é uma aplicação lógica programável acessível utilizando-se protocolos padrões da Internet, combinando os melhores aspectos de desenvolvimento baseado em componentes e sendo acessíveis via protocolos web utilizando-se de formatos de dados padrões, por exemplo, XML [4].

Paralelamente ao advento da grande rede, o avanço da micro eletrônica permitiu, dentre outros, o desenvolvimento de pequenos equipamentos com limitada capacidade computacional, chamados de PDAs, também denominados de *Handhelds*. Por serem pequenos, do tamanho do bolso de uma camisa, são extremamente versáteis devido a sua capacidade de mobilidade. Podem ser usados a qualquer momento praticamente independentemente da localização ou ambiente onde seus usuários se encontram.

O sucesso comercial dos PDAs pode ser verificado pela Tabela 1.1, contendo estimativas e projeções de diferentes empresas de consultoria. Com isso, cada vez mais a capacidade de processamento e de armazenamento destes dispositivos aumenta. Ao ponto de já possuírem várias das funcionalidades de um computador pessoal.

Tabela 1.1: Comparativo de estimativas de vendas de PDAs [5]

	2000	***2001***	***2002***	***2003***	***2004***	***2005***
ABN AMRO	10,7	11,9	15	21,3	29,2	-
CSFB Tecnology Group	9,4	12,2	17,7	24,4	31,8	-
eTForecasts	12,2	-	24,4	-	35,6	-
Gartner Dataquest	10,9	12,3	16	23,1	31,6	39,2
International Data Corporation (IDC)	13,6	-	-	-	-	70,9
UBS Warburg	10,3	17,7	25,9	-	53,7	-
Aberdeen Group	9	11,8	16,8	23,1	30,7	39,3

Não só os PDAs são advenientes do avanço da micro eletrônica. Esta também possibilitou o surgimento das redes de telefonia celular, que permite aos seus usuários se comunicarem através de telefones com total mobilidade nas áreas de cobertura. A disseminação da telefonia celular é tão grande que o telefone celular passou de um produto de luxo para se tornar um item essencial na vida moderna. Só no Brasil há quase 42 milhões de telefone celulares [6].

O que ocorre no mundo da telefonia móvel hoje é a convergência tecnológica entre os PDAs e os telefones celulares, com o aparecimento de um novo produto: os *smartphones* (telefones inteligentes). Tais dispositivos eletrônicos possuem as qualidades de um PDA e de um telefone celular em um mesmo produto, ou seja, possuem a capacidade de mobilidade, de armazenamento e processamento de um PDA associado às facilidades de comunicação móvel via rede de telefonia celular.

Um novo e crescente mercado vem à tona a fim de atender os usuários dessa nova tecnologia, ávidos por diferentes tipos de serviços e pela possibilidade de poderem ter acesso à Internet a partir de seus *smartphones*, com toda a mobilidade possível. Um relatório emitido pela ABI indica que a comercialização de s*martphones* (que denominam de PDAs conectados) já está suplantando a de PDAs (ver Figura 1.1).

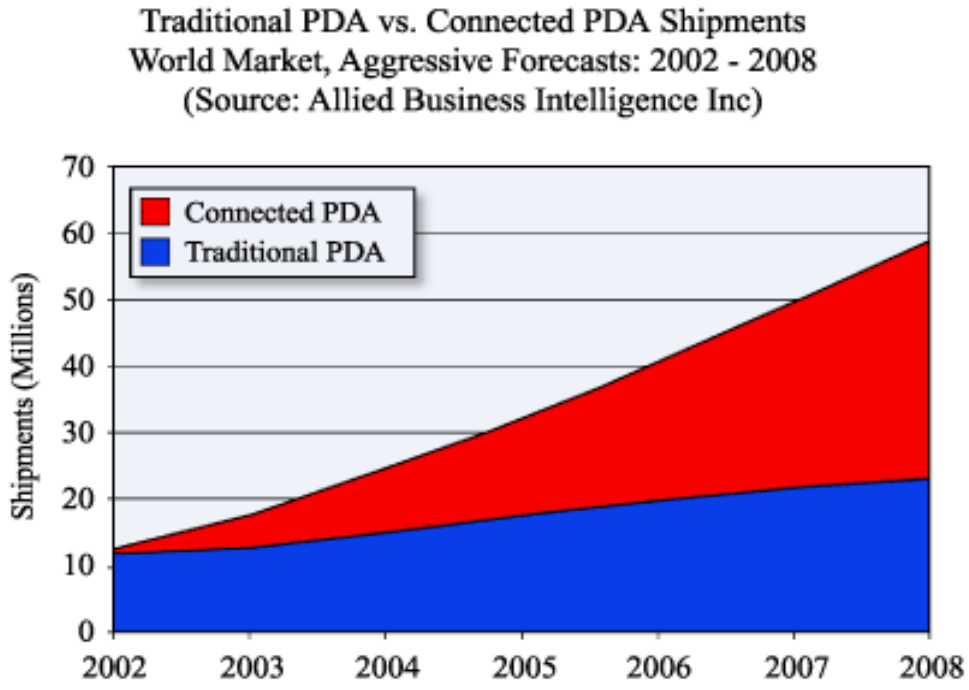


Figura 1.1: Venda de PDAs *Smartphones* no mundo [7]

1.2 Objetivos e Estrutura do Trabalho

O objetivo principal deste trabalho é a realização de um estudo de caso de fornecimento de serviços a usuários de dispositivos móveis - tais como *smartphones* - utilizando-se de *Web Services* para implementar um sistema distribuído. Uma das preocupações deste estudo é garantir o mínimo de qualidade e disponibilidade, supondo uma grande utilização deste sistema.

Primeiramente é introduzido o conceito de computação móvel, além de serem abordadas de maneira sucinta e vinculada (tanto ao conceito de computação móvel, quanto ao estudo de caso) as redes de telefonia celular, os PDAs e, por fim, os *smartphones*.

Na seqüência descreve-se o que é um *Web Service* e as tecnologias inerentes ao mesmo, tais como XML, SOAP, WSDL e UDDI. Tais tecnologias foram utilizadas para a implementação dos serviços abordados neste estudo.

Fala-se, então, sobre *clusters* visando entender como funciona esta alternativa para a montagem de um ambiente de alto processamento e tolerante a falhas.

Ainda sobre *clusters*, explana-se sobre o OSCAR: uma ferramenta que facilita enormemente a instalação, configuração e gerenciamento de um ambiente baseado em *clusters*. Também é abordado o tema de balanceamento de carga com a intenção de demonstrar as possibilidades para se garantir alto desempenho e alta disponibilidade na oferta de serviços via *Web Services*.

O estudo de caso realizado é comentado em seguida, onde descreve-se a finalidade (e utilidade) do sistema e as ferramentas escolhidas.

Após o estudo, segue a parte final deste trabalho onde disserta-se sobre os problemas e dificuldades encontradas para a implementação do serviço. São relacionadas sugestões para trabalhos futuros, finalizando com uma conclusão geral sobre o trabalho.

2 COMPUTAÇÃO MÓVEL

A computação móvel surge como uma quarta revolução na computação, antecedida pelos grandes centros de processamento de dados da década de sessenta, o surgimento dos terminais nos anos setenta, e as redes de computadores da década de oitenta. O novo paradigma permite que usuários desse ambiente tenham acesso a serviços independentemente de onde estão localizados, e o mais importante, de mudanças de localização, ou seja, mobilidade [8].

Para exercer este modelo de computação, faz-se necessário ter-se um dispositivo computacional móvel com capacidade de se comunicar a outros equipamentos através de uma estrutura de rede que não limite a mobilidade do mesmo. Para tanto, a utilização de uma estrutura fixa baseada em meio guiado de rede para interconexão de dispositivos é inviável. A alternativa é o uso de uma estrutura para transmissão de informações via um meio não-guiado, tal como radio-freqüência.

Além disso, também necessita-se de um dispositivo computacional com o mínimo de capacidade de processamento e armazenamento e que possa se comunicar com outros dispositivos.

2.1 Redes sem Fio

Como exemplo de tecnologias sem fio que surgiram nos últimos anos para atender as necessidades de computação distribuídas, pode-se citar o *Bluetooth* e o padrão IEEE 802.11.

A tecnologia sem fio *Bluetooth* é um padrão de fato e uma especificação para enlaces entre dispositivos móveis, tais como computadores pessoais, assistentes digitais pessoais, telefones celulares e outros dispositivos portáteis de baixo custo, usando ondas de rádio de curto alcance [1]. É uma tecnologia de rede sem fio ad-hoc (sem estrutura definida), indicada para as necessidade de interconexão de equipamentos em pequenas áreas. O BSIG é um grupo industrial formado várias empresas, cujo objetivo é suportar e desenvolver esta tecnologia.

A especificação IEEE 802.11 surgiu da necessidade de se padronizar os serviços de redes locais sem fio. Ela define duas topologias de rede: redes de conjunto de serviços básicos independentes (*IBSS Networks*) e redes de conjuntos de serviços estendidos (*ESS Networks)* [1]. O primeiro compreende uma estrutura de rede ad-hoc, enquanto o segundo refere-se a uma estrutura híbrida com pontos de acesso interconectados via uma rede *Ethernet*. Também utiliza radio-freqüência como meio de transmissão através dos pontos de acesso.

Ambas tecnologias têm sua aplicação no mundo da computação distribuída. Porém *Bluetooth* possui limitada área de cobertura e o uso da especificação IEEE 802.11 não é difundida como são as redes de telefonia celular. Apesar de vários modelos de *smartphones* serem capazes de se conectar a redes *Bluetooth* e IEEE 802.11, estes utilizam as redes de comunicação móvel celular como principal meio para a transmissão de dados.

2.2 Telefonia Celular

A telefonia celular está difundida em todo o mundo, e também no Brasil. É um mercado crescente, com uma incrível demanda de serviços e concorrência forte entre operadoras. Este pequeno aparelho de comunicação deixou de ser um item de luxo ou supérfluo, para ser uma necessidade da vida moderna.

2.2.1 História

O surgimento dos microprocessadores permitiu o grande impulso desenvolvedor na telefonia celular. Credita-se a Martin Cooper, vice-presidente da Motorola, a invenção do telefone celular, cuja tecnologia utiliza-se atualmente (o que ele construiu foi um pequeno rádio telefone que podia ser carregado por uma pessoa). Ele fez a primeira ligação em 1973, de uma esquina em Nova Iorque, usando a base instalada no topo de um edifício que ficava próximo. Esta ligação foi para um amigo na AT&T. A Motorola tornou pública sua tecnologia em 1983, depois de dez anos, cinco gerações de chips e noventa milhões de dólares em pesquisa.

Os verdadeiros inventores da telefonia celular foram pesquisadores do *Bell Labs*, que desenvolveram o conceito das pequenas células e implementaram o conceito de *switch*, ou seja, a mudança automática de uma célula a outra, à medida que o aparelho telefônico se desloca. Por último, a explosão de tecnologia celular atual foi possível, como mencionado anteriormente, graças aos microprocessadores e aos pequenos chips que foram desenvolvidos pela Intel, ainda que Motorola e AMD tenham dado também importantes contribuições nesta área [9].

A primeira rede de telefonia celular do Brasil foi implantada no estado do Rio de Janeiro em 1990. Entrou em funcionamento do dia trinta de dezembro do mesmo ano, com capacidade para dez mil terminais (telefones celulares). Porém o serviço era caro e mesmo começou a se expandir a partir de 1992 [10].

2.2.2 Telefonia Celular no Brasil

As principais operadoras de telefonia celular no país são: OI-Telemar, Claro, TIM, VIVO, Amazônia Celular e Telemig. Segundo levantamento da ANATEL feito a partir dos relatórios trimestrais das operadoras, há quase quarenta e dois milhões de telefones celulares em funcionamento no Brasil [6]. Não muito menos do que o número de linhas telefônicas fixas instaladas, 49.423.482 linhas (terceiro trimestre de 2003) [6].

Tabela 2.1: Dados da ANATEL sobre a telefonia celular no Brasil [6]

DADOS RELEVANTES DO SMC/SMP (Posição Outubro/2003)					
Plano/Serviço	SMC	SMP	Total	Participação (%)	Densidade (acessos/100 hab.)
Pós-Pago	1.176.663	9.571.927	10.748.590	25,66	6,08
Pré-Pago	2.342.721	28.796.608	31.139.329	74,34	17,61
Total	3.519.384	38.368.535	41.887.919	100,00	23,68
Acessos Móveis por Banda					
A			24.564.098	58,64	13,89
B			12.467.783	29,76	7,05
D			4.304.318	10,28	2,43
E			551.720	1,32	0,31
Total			41.887.919	100,00	23,68
Acessos Móveis por Tecnologia					
AMPS			751.859	1,79	0,43
TDMA			23.619.043	56,39	13,35
CDMA			12.613.673	30,11	7,13
GSM			4.903.344	11,71	2,77
Total			41.887.919	100,00	23,68

As tecnologias para redes de telefonia celular autorizadas pela ANATEL a serem utilizadas no Brasil são: AMPS, TDMA, CDMA e GSM.

2.2.3 Princípios Básicos das Redes de Telefonia Celular

As redes de telefonia celular são compostas por áreas de comunicação denominadas de células Ao centro de cada célula há uma estação base, que transmite sinais analógicos para usuários pertencentes àquela célula. Se um usuário móvel passa do limite de uma célula, o sinal precisa ser passado para a estação base da célula que o usuário entrou. A transferência de sinais a partir de uma célula para outra é chamado *handoff*. O *handoff* permite aos usuários móveis continuar a se comunicar sem perda de conexão ou interrupções de rede (ver Figura 2.1).

A seguir descreve-se sucintamente cada um dos tipos de rede de telefonia celular, focando-se apenas nas taxas de transmissão de dados de cada uma.

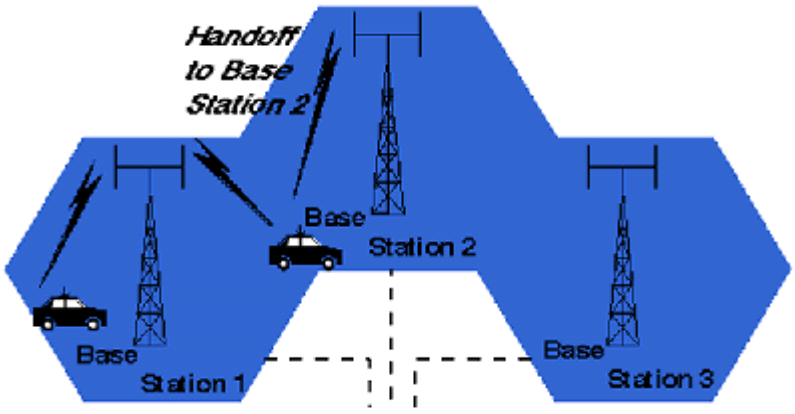


Figura 2.1: *Handoff* de um usuário [11]

2.2.4 Redes G1

A primeira geração de redes de telefonia celular utiliza transmissão analógica de sinais através de ondas de rádio. No Brasil, a tecnologia adotada foi a AMPS.

Neste tipo de tecnologia, um dispositivo móvel envia sinais na forma de ondas para uma estação base onde eles são processados para determinar o próximo destino do sinal. Uma vez o destino determinado, o sinal é reconstituído o mais fielmente possível em sua forma de onda original pela estação base. O sinal analógico recebido pelo usuário fim pode se aproximar muito ao da transmissão original mas raramente o reproduz fielmente. Diferenças perceptíveis em qualidade e forma ocorrem devido a erros na recriação da onda. Destruição de sinal (o sinal é perdido ou danificado durante a transmissão), problemas de tradução e interferência ameaçam transmissões analógicas mais gravemente que transmissões digitais [11].

A transmissão de dados só é possível via modem, a uma taxa efetiva de 1,2 kbits/s [12].

2.2.5 Redes G2

As redes de segunda geração transmitem sinais digitais, diferentemente das redes de primeira geração. Esta é a principal diferença entre as duas. Os *bits* que compõem o sinal digital enviado permitem que a estação base duplique o mesmo e possa transmitir para seu próximo destino. Apesar do novo sinal ser praticamente uma réplica do original, erros de tradução podem ocorrer, porém são bem reduzidos em comparação com os sistemas analógicos [11].

No Brasil, as tecnologias TDMA, GSM e CDMA de segunda geração são as autorizadas para utilização das operadoras de telefonia celular.

A especificação TDMA para redes G2 é uma versão digital do protocolo AMPS. Um fator essencial na mudança do sistema AMPS para o sistema TDMA digital nos países que adotaram o AMPS é o aproveitamento da estrutura já montada e largamente utilizada. Possui capacidade de transmissão de dados a uma taxa efetiva de 4,8 kbits/s [11].

O CDMA é um dos padrões mais populares, apenas atrás do número de usuários do TDMA. Possui um princípio de funcionamento diferente do TDMA, necessitando de menos potência para transmissão de sinais o que permite maior autonomia de funcionamento dos telefones celulares [12]. A taxa efetiva de transmissão de dados do CDMA é de 9,6 kbits/s.

Por fim, o GSM tem como base a tecnologia TDMA e surgiu primeiramente na Europa, devido à necessidade de padronizar um sistema celular digital que cobrisse todo o território do continente [12]. A taxa efetiva de transmissão de dados do GSM varia entre 9,6 kbits/s e 14,4 kbits/s [11].

2.2.6 Redes G2.5

As redes de geração 2.5 são intermediárias entre as G2 e G3, e incluem redes com taxas de transmissão de 100 kbits/s [11]. Abaixo seguem as diferentes tecnologias e especificações para as redes G2:

- HSCSD: é uma especificação projetada para melhorar as redes GSM e aumentar a taxa de transmissão de dados para 115 kbits/s, utilizando-se de comutação de circuitos para transmissão de dados [12];
- GPRS: outra tecnologia projetada para melhorar as redes GSM. Possui como principal característica o fato de o seu usuário estar conectado permanentemente à rede, e não somente quando há demanda. Sua taxa de transmissão de dados é de até 196 kbits/s;
- CDMA2000: visa estender o CDMA para redes G3, e suporta diferentes tecnologias baseadas no conjunto de especificações CdmaOne como também GSM e TDMA. Possui taxa de transmissão de dados de até 114 kbits/s.

2.2.7 Redes G3

As redes G3, de terceira geração, são caracterizadas pela capacidade de transmissão de dados a uma taxa acima de 300 kbits/s. Abaixo, citam-se algumas das tecnologias atualmente disponíveis:

- EDGE: funciona sobre redes GSM em associação com GPRS e TDMA. Também provê conectividade persistente e suporta transmissão de dados a uma taxa de até 384 kbits/s [11];
- W-CDMA: expande a capacidade das redes CDMA e aumenta as taxas de transmissão para até 2 Mbits/s [11].

2.3 PDA

Os PDAs são dispositivos computacionais de tamanho de uma carteira que possuem memória e capacidade de processamento limitado. Várias são as empresas que produzem e comercializam este tipo de equipamento, dentre elas: PalmOne, Sony, Compaq, Apple, Garmin, Alpha Smar e Fossil.

Atualmente a grande disputa no mercado de PDAs é a dos sistemas operacionais para dispositivos móveis. Os dois maiores concorrentes são a

plataforma Windows CE [13] desenvolvida pela própria Microsoft e a plataforma Palm OS, desenvolvida pela PalmSource [14].

2.3.1 História

O primeiro PDA lançado no mercado mundial foi o Psion I em 1984, produzido pela empresa britânica Psion. Era semelhante a uma calculadora de bolso, possuía um processador de oito *bits* e um LCD com capacidade para exibir dezesseis caracteres [15].

A Apple, em 1993, comercializou sua versão de um PDA: o Newton. No entanto, em 1998 toda a linha de PDAs da Apple foi descontinuada [16]. Recentemente, a mesma voltou ao mercado de PDAs com a linha iPod [17].

Já em 1996 a Palm Inc lançou os PDAs Pilot 1000 e Pilot 5000, que lideraram uma revolução na computação móvel [18], devido ao volume de vendas alcançado. Como conseqüência outras empresas também começaram a comercializar dispositivos computacionais móveis, causando a expansão deste mercado.

2.3.2 Windows CE

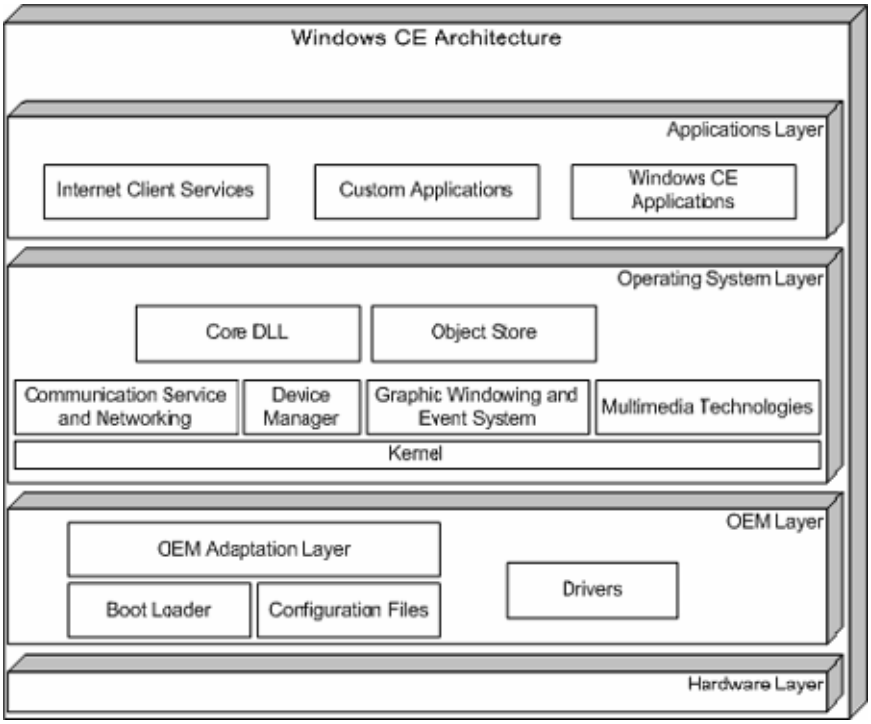
A Microsoft desenvolveu uma versão de sua família de sistemas operacionais Windows para funcionar em sistemas embutidos, tais como PDAs, *smartphones* e terminais de ponto de venda, cujo nome é Windows *CE*.

Já que minimização dos requisitos de memória é o maior objetivo de um projeto, o Windows CE é construído a partir de um conjunto discreto de módulos e sub módulos, ou componentes. Cada um destes módulos provê total ou parcial suporte dos relevantes recursos de um sistema operacional, e podem ser adaptados às necessidades de uma aplicação embutida específica [19].

A Figura 2.2 ilustra o nível de modularidade da arquitetura do Windows CE, especificamente em sua versão 4.20.

As camadas *Hardware Layer* e *OEM Layer* abstraem as especificidades de um determinado dispositivo (um PDA, por exemplo) e fornecem um interface genérica para que as camadas superiores acessem seus serviços. Com isso, a camada do sistema operacional (*Operating System Layer*) é totalmente portável para diferentes plataformas de dispositivos.

Na camada de sistema operacional, temos o núcleo do sistema operacional (*kernel*) do Microsoft Windows CE. Este fornece as funcionalidades básicas do sistema operacional para qualquer dispositivo em que funcione [19]. Suporta memória virtual, é multi-tarefa, possui suporte a *threads* (vários fluxos de execução em um processo) e também suporte a grande parte da API das versões de Windows para servidores e computadores pessoais. No mais, desta camada ainda fazem parte alguns componentes responsáveis pelos serviços de comunicação e rede, gerenciamento do dispositivo, gerenciamento de janelas gráficas, armazenamento de dados e suporte à multimídia.



Por fim as aplicações utilizadas pelos usuários funcionam acima da camada de sistema operacional, utilizando-se dos serviços fornecidos pela mesma.

2.3.3 Palm OS

Pelas palavras da própria PalmSource a versão Palm OS 5 é a “fundação do futuro da computação móvel”. Ele é mais do que um sistema operacional, é uma plataforma desenvolvida pela PalmSource para computação móvel.

O sistema operacional desenvolvido pela PalmSource suporta diferentes tipos de hardwares, segurança de informações, tem capacidade para lidar com aplicações multimídia e permite conectividade utilizando tecnologias de comunicação sem fio com a adição de 802.11b (no Palm OS 5.0), assim como as já suportadas tecnologias: *Bluetooth*, GSM, CDMA e redes G2.5 e G3 [20].

A versão 5 possui uma nova arquitetura, porém é compatível com os programas aplicativos desenvolvidos para versões anteriores do Palm OS. É um sistema operacional de 32 bits que roda nativamente em processadores compatíveis com ARM e permite a execução de programas escritos para a família de processadores Motorola 68000 (*DragonBall*) através da camada PACE. A figura 4 permite a visualização da estrutura do Palm OS.

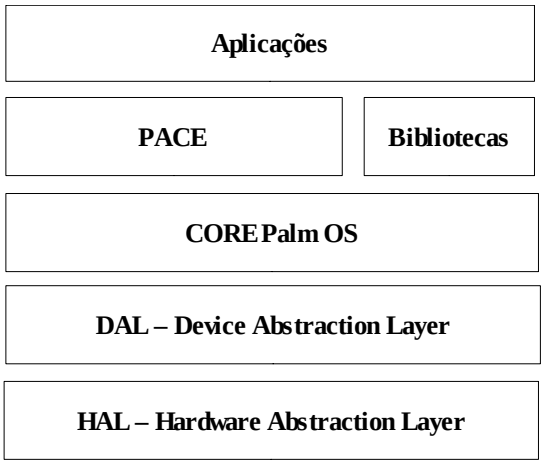


Figura 2.3: Estrutura do Palm OS 5 [20]

2.3.3.1 HAL/DAL

As camadas HAL ou camada de abstração de hardware e DAL ou camada de abstração de dispositivo dão uma maior versatilidade ao Palm OS, permitindo que o mesmo suporte vários tipos de diferentes hardware, sejam estes PDAs, *smartphones* ou outros dispositivos eletrônicos. Elas disponibilizam serviços básicos de acesso ao hardware, abstraindo particularidades do mesmo para o núcleo do sistema operacional. Por isso, na necessidade de se portar o Palm OS 5 para uma nova plataforma, apenas as camadas HAL e DAL precisam ser adaptadas.

2.3.3.2 O CORE do Palm OS

É a camada onde localiza-se o núcleo do sistema operacional. O Palm OS possui um *microkernel* multitarefa preemptivo que provê um gerenciamento básico de tarefas (processos). A maior parte das aplicações não acessa diretamente os serviços oferecidos pelo núcleo [21].

Nesta versão do Palm, somente há uma aplicação com interface de usuário sendo executada num determinado momento. O UIAS é responsável por gerenciar a aplicação com interface de usuário corrente. O mesmo inicia a execução de um programa como se esta fosse uma subrotina, e não obtém o controle do ambiente enquanto o programa não parar de funcionar [21]. Por essa razão, é impossível solicitar a execução de uma nova aplicação com interface gráfica enquanto há outra funcionando. O que ocorre quando troca-se de uma aplicação para outra é a suspensão da execução do programa corrente.

Porém, é possível a uma aplicação iniciar outro processo através dos serviços do núcleo. Os programas de sistema do Palm, como o *HotSync*, utilizam esses recurso. O *HotSync* cria um processo dedicado para realizar a comunicação serial e configura esse processo com uma prioridade de execução menor que a sua [21]. Com isso, ainda é possível utilizar o Palm mesmo quando seus dados estão sendo sincronizados com os de um microcomputador. A PalmSource, para a versão 5 do PalmOS, definiu que

esse recurso não está disponível para desenvolvedores de aplicações para o Palm.

2.3.3.3 PACE

Cada aplicação que funciona no Palm OS é executada sobre a camada PACE. Ela emula um processador da família Motorola 68000. Apesar disso, a grande maioria das aplicações é beneficiada pelo aumento de performance ganho pela plataforma ARM, pois utilizam intensivamente a API do Palm OS que roda nativa nativamente na plataforma ARM [21].

Para aquelas que necessitam de processamento intensivo, é possível escrever subrotinas ARM que rodam nativamente na nova arquitetura [21]. Essas rotinas entram em execução fora da camada PACE, bastando ao programador utilizar a chamada de sistema PceNativeCall. A utilização de tais subrotinas é somente recomendada em caso de funções que realmente podem se beneficiar, tais como algoritmos de encriptação e compactação [21].

2.3.3.4 Bibliotecas

As bibliotecas do Palm OS e de terceiros permitem aos programadores de aplicações utilizarem a maior parte dos recursos disponíveis pelo sistema operacional e pelo dispositivo.

São fornecidas bibliotecas de objetos gráficos para construção de interfaces gráficas, para utilização de protocolo TCP/IP, sincronização de dados, comunicação sem fio, e muitas outras.

Tais bibliotecas estão fora da camada de emulação PACE, o que lhes permite ter total ganho de desempenho ao funcionarem na plataforma ARM. Em razão disso, as aplicações que rodam acima da camada PACE também aproveitam esse aumento.

2.3.3.5 Aplicações

Esta é a camada onde os programas aplicativos funcionam. Juntamente com o Palm OS são fornecidos um conjunto de programas básicos para serem utilizados, por exemplo:

- *Address*: Para armazenamento de endereços e contatos;
- *Calc*: simula uma calculadora simples;
- *Date Book*: Agenda eletrônica de compromissos;
- *MemoPad*: Escrita de memorandos;
- *ToDo List:* Lista de tarefas;
- SMS: Envio de mensagens via SMS (se suportado pelo dispositivo);
- *HotSync*: Realizar sincronização de dados entre um computador desktop e o dispositivo.

2.4 Smartphone

O *Smartphone* é um produto resultante da convergência de todas as tecnologias citadas neste capítulo e que viabilizam a computação móvel. Com ele é possível acessar quaisquer serviços através das redes de telefonia celular, e tendo a mesma capacidade dos PDAs também possibilita a execução de aplicações mais complexas, ao contrário de um telefone celular.

Ele combina a funcionalidade dos PDAs, PCs *wireless*, telefones e mesmo câmeras digitais em um único aparelho, tornando-se uma poderosa ferramenta para negócios.

Com a popularização dos telefones inteligentes, a revolução da computação móvel [10] é iminente.

2.4.1 O que são

Smartphones são telefones móveis digitais que podem enviar e receber voz, dados e vídeo, além de armazenar informação e executar programas. As funções variam de telefone para telefone, mas todos podem conectar-se à Internet para acesso à *web* e *e-mail.* Um *smartphone* consolida a tecnologia de telefonia móvel com a de um PDA para criar uma poderosa ferramente de comunicação.

Alguns dos mais modernos *smartphones* podem incorporar capacidades G3.

2.4.2 Benefícios

Já que a maior parte das suas funcionalidades é embutida, os *smartphones* podem fazer coisas muito mais rapidamente que seus precursores: o telefone celular e o PDA.

Os *smartphones* podem funcionar como organizadores pessoais, com diários eletrônicos, lista de contato e lembretes e assim como um um PDA, pode-se utilizar um *smartphone* para fazer anotações, ver e editar compromissos, contatos e documentos, tudo enquanto o usuário se movimenta.

Cada vez mais e mais serviços estão sendo disponibilizados em *smartphones*, indo de acesso a mapas e caminhos a transmissões de televisão com cobertura de notícias ou previsões de tempo e também a informações sobre trânsito e alertas sobre horários – o que significa que um negócio pode sempre estar um passo à frente do que está acontecendo.

Pode-se conectar-se à Internet para navegar ou ler *e-mails* a qualquer momento, ou ligar-se a uma rede de computadores para acesso a dados importantes, onde quer que o usuário esteja. Em um *smartphone o* acesso a Internet é mais rápido do que nos dispositivos móveis anteriores, tornando mais fácil acessar *e-mail* e informação da *web*.

Além disto, consideravelmente mais dados podem ser recebidos e transmitidos via um *smartphone*, tais como grandes anexos em *e-mails* ou arquivos de *websites*. Aparelhos móveis anteriores podiam apenas gerenciar pequenos *e-mails* sem anexos.

Finalmente, eles apresentam uma maior funcionalidade, já que muitos modelos oferecem câmeras digitais embutidas com funcionalidades para bater e enviar fotos. Com isto, trabalhadores remotos podem instantâneamente fotografar e enviar imagens ou vídeos de todos os produtos de um fornecedor para inspeções, por exemplo. *Smartphones* podem oferecer aos usuários uma nova maneira de marketing com seus consumidores, utilizando-se de MMS para incluir animações, imagens e música em uma mensagem.

2.4.3 Funcionamento

Externamente os *smartphones* funcionam de forma muito semelhante aos telefones celulares atuais. Os teclados são uma “mistura” entre os encontrados em telefones celulares e PDAs e os dispositivos em si possuem o mesmo tamanho e peso dos telefones celulares, além de sererm normalmente mais iluminados e menores que os PDAs. Uma grande diferença é que eles podem também acessar conteúdo personalizado tal como J2ME. O J2ME é acessível via *applet* Java em um *smartphone* e fornece um acesso rápido e relativamente barato a um grande número de aplicativos úteis para negócios, tais como processadores de texto, planilhas eletrônicas, etc.

Os *smartphones* oferecem, então, a mobilidade de telefone celular com a multifuncionalidade de um PDA. Modelos mais avançados possuem *Blutetooth*, conectando-se sem fio e com segurança com outros dispositivos *Bluetooth* em curtas distâncias para a criação de redes locais. A conectividade a outras tecnologias pode ser conseguida através de infra-vermelho e, em muitos modelos, estão disponíveis cartões de memória de várias capacidades.

Eles parecem ser, definitivamente, um passo à frente. Os problemas associados aos PDAs foram solucionados para sua incorporação nos

smartphones. O número de aplicações disponíveis está crescendo, a tecnologia é estável e confiável e a percepção dos usuários profissionais e comuns sobre isto está crescendo.

2.5 M-Commerce

Após o advento e da expansão do comércio eletrônico, a chegada dos dispositivos móveis com acesso à Internet vem adicionar novos desafios ao comércio. O conceito de *m-commerce* (*mobile commerce*) que já faz parte do vocabulário de europeus, americanos e japoneses, torna-se a cada uma realidade no Brasil e no mundo. Estudos realizados recentemente no âmbito desta nova modalidade de comércio indicam que, apesar das transformações tecnológicas nas comunicações móveis estarem ainda em fase de desenvolvimento, a sua evolução e consolidação será mais acelerada que a da Internet [22].

Mobilidade de comércio eletrônico, o *m-commerce* se diferencia do comércio eletrônico convencional, porque é realizado por meio de telefones celulares ou outros aparelhos portáteis, como laptops, computadores de mão (terminais sem fio), ao invés de equipamentos fixos. Pode-se dizer que é uma versão móvel do *e-commerce*, sendo um sub-grupo deste.

O *m-commerce* pode ser entendido como a combinação natural que resulta da evolução das comunicações móveis, capazes de transpor progressivamente para a rede sem fio produtos, serviços e aplicações da Internet, e levando à criação de outros serviços e produtos que resultarão das características e potencialidades únicas do ambiente sem fio. Enfim, define-se como a venda de produtos e serviços ao consumidor através de dispositivos móveis.

Mas o *m-commerce* é muito mais do que comprar um produto através do telefone. Os serviços incluem: serviços financeiros, a compra de bilhetes impressos, informações sobre viagens, ações, meteorologia, entre

outros, tudo isto representando as primeiras oportunidades de implantar serviços caracterizados como comércio móvel.

O desenvolvimento tecnológico das redes móveis já permite uma transmissão mais rápida de maiores volumes de informação para os aparelhos novos que estão surgindo no mercado. As tecnologias desenvolvidas possibilitam, por exemplo, funções multimídia, a realização de videoconferência, serviços de localização geográfica, o acesso permanente à Internet e a possibilidade de comprar um bem ou serviço a partir de qualquer lugar, a qualquer hora, sem que para tal o consumidor necessite se deslocar à loja.

2.6 O Mercado

O *m-commerce* encontra-se em constante expansão não somente por facilitar as transações comerciais, mas por ampliar as oportunidades de negócios. Em vista disso, empresas em todo o mundo demonstram um grande interesse neste inovador ramo do comércio, e as que deixam transparecer maior interesse são aquelas que já possuem algum tipo de comércio eletrônico em atividade, pois não querem ficar para trás neste desenvolvimento da tecnologia móvel. Mas não é de hoje que os interesses e investimentos em Internet móvel e nesta nova forma de se fazer comércio apareceram, muitas empresas principalmente as americanas, européias e japonesas, há alguns anos, estudam e já vêm colocando em prática a utilização do comércio móvel.

Na Europa existe uma variedade de livros e publicações de revistas que tratam do *m-commerce*. As empresas de telecomunicações acreditam tanto no crescimento da internet móvel que na Inglaterra o leilão das licenças da tecnologia G3 rendeu 35 bilhões de dólares ao governo, isso somente para a utilização do espectro de rádio [23]. Mas todo este investimento não é à toa, eles sabem que é crescente o número de usuários de telefone celular com acesso à Internet e acreditam que o *m-commerce* irá cobrir certos custos e trazer lucros.

3 WEB SERVICES

3.1 Introdução

Um *Web Service* é uma aplicação lógica programável acessível utilizando-se protocolos padrões da Internet, combinam os melhores aspectos de desenvolvimento baseado em componentes e são acessíveis via protocolos web utilizando-se de formatos de dados padrões, por exemplo, XML [4].

Web Services tem surgido como um mecanismo poderoso para a integração da diferentes tipos de sistemas, podendo ser adotados gradativamente com risco e custo reduzidos. Atualmente as empresas utilizam *Web Services* para integração de aplicações ponto a ponto, para reutilização de componentes e para conectar-se a parceiros de negócios ou clientes.

Historicamente falando *Web Services* representa a convergência entre SOA [24] e a Web. As SOAs evoluíram nos últimos dez anos para suportar alto desempenho, escalabilidade, confiabilidade e disponibilidade. Para alcançar o melhor desempenho as aplicações são projetadas como serviços que são executados em um cluster de servidores de aplicação centralizado.

Um serviço é uma aplicação que pode ser acessada através de uma interface programável. No passado os clientes acessavam tais serviços utilizando protocolos de computação distribuída, tais como DCOM [25], CORBA [26] ou RMI [27], os quais são excelentes para a construção de uma aplicação específica, mas reduzem a flexibilidade do sistema na medida que

limitam a reusabilidade de serviços individuais. Cada um destes protocolos está limitado por dependências de implementações de cada fabricante, plataformas, linguagens ou esquemas de codificação de dados que reduzem severamente a interoperabilidade entre eles. Além disso, nenhum deles opera efetivamente através da *Web*.

A arquitetura de *Web Services* toma vantagem de todos os benefícios da arquitetura orientada a serviços e a combina com o suporte da *Web* e sua comunicação universal, a qual utiliza-se de protocolos completamente independentes de fabricante, plataforma e linguagem, resultando numa arquitetura que suporta acesso via *Web*, fácil integração e reusabilidade de serviços.

Visto o exposto anteriormente qualquer tipo de aplicação pode ser disponibilizada como um *Web Service*, sendo aplicável a qualquer tipo de ambiente *Web*: Internet, *intranet* ou *extranet*. *Web Services* suportam interações entre empresas, empresas e clientes, entre departamentos ou ponto a ponto e seu cliente pode ser um usuário acessando o serviço através de seu desktop ou de um dispositivo *wireless*, uma aplicação ou outro *Web Service*, suportando os *frameworks* de segurança disponíveis atualmente.

3.2 Características

- acessível através da *Web*, comunicando-se através de protocolos independentes de plataforma e linguagem, permitindo uma fácil integração entre ambientes heterogêneos;
- disponibiliza uma interface que pode ser “chamada” de outro programa. Esta interface de programação programa a programa pode ser invocada de qualquer tipo de aplicação cliente ou serviço. Ele atua como uma ligação entre a *Web* e a aplicação lógica real que implementa o serviço;
- um *Web Service* é registrado e pode ser localizado através de um *Web Service Registry* [28]. O registro permite que os clientes do serviço encontre os que mais se adequem às suas necessidades;

- *Web Service* suporta conexões sem controle entre sistemas. Eles se comunicam enviando mensagens uns para os outros. Sua interface adiciona uma camada de abstração ao ambiente que torna as conexões flexíveis e adaptáveis.

3.3 Tecnologias

Um *Web Service* pode ser desenvolvido utilizando qualquer linguagem de programação e pode ser utilizado em qualquer plataforma, além de poder se intercomunicar já que todos “falam a mesma língua”, o XML, utilizando-o para descrever suas interfaces e para codificar suas mensagens. *Web Services* baseados em XML comunicam-se através de protocolos *Web* padrões utilizando interfaces e mensagens XML, as quais podem ser interpretadas por qualquer aplicação.

Entretando o XML por si só não garante uma comunicação sem esforços. As aplicações precisam de formatos e protocolos padrões que os permitam interpretar corretamente o XML.

Neste sentido três tecnologias baseadas em XML surgiram como padrões de fato para *Web Services*:

- **SOAP** – define um protocolo de comunicação padrão para *Web Services*.
- **WSDL** – define um mecanismo padrão para descrever um *Web Service*.
- **UDDI** – fornece um mecanismo padrão para registrar e encontrar *Web Services*.

3.3.1 XML

XML é um formato texto simples e flexível, derivado de SGML [4]. Foi projetado para permitir a troca de dados entre diferentes instituições, através da Internet. Devido à sua adaptabilidade, passou a ser utilizado para trocar informações complexas que não eram adequadamente expressas em HTML, como equações matemáticas e dados geográficos.

XML é uma metalinguagem descritiva, isto é, é uma linguagem que permite descrever, definir novas linguagens, cada uma delas voltada especificamente para um tipo de informação. Isto é realizado através da criação de um DTD (conforme item 3.3.1.1). A partir dela, um grande conjunto de novas especificações foram criadas, procurando acrescentar novas facilidades ao XML: *XML Schema*, XSLT, DOM e SAX. A especificação completa de XML e demais tecnologias associadas estão disponíveis em [4].

Assim como é importante definir adequadamente XML, é também importante deixar claro o que ela não é: uma linguagem para formatação de textos. Neste sentido, XML não é substituta para HTML, pois não possui *tags* (marcas), por exemplo, para negrito ou itálico, como o HTML. XML permite trocar dados, com a sua semântica, entre instituições, de modo que é possível desenvolver programas que os interpretem adequadamente e de maneira automática, sem a necessidade de leitura por profissionais.

3.3.1.1 DTD

Um DTD determina as partes de um documento XML descrevendo a sua estrutura interna. Basicamente, um DTD é um conjunto de regras que definem os diversos elementos que devem estar presentes em um documento XML, a hierarquia entre os elementos e o número de vezes em que devem aparecer [29]. Assim, é possível a um analisador sintático reconhecer se um documento está sintaticamente correto ou não, o que pode ser muito importante como validação de dados.

É importante porque descreve qual a estrutura esperada para o documento. O DTD pode ser resultado da concordância de todos interessados, ou imposição por algum órgão com poder regulatório, ou, ainda, uma combinação de ambos.

3.3.1.1.1 Definição de um DTD

Um DTD é construído a partir de três conceitos básicos: elementos, atributos e entidades [29] [30]. Elementos são as *tags* a serem utilizadas no

arquivo XML. Devem estar organizadas em uma estrutura de árvore hierárquica, sendo declaradas a partir da raiz da árvore.

Os atributos ajudam a descrever exatamente os elementos, o tipo de informações que devem ser inseridas neles e a ordem na qual as informações devem ser colocadas. Os atributos devem ser posicionados após a declaração do elemento.

A principal diferença entre elementos e atributos é que os elementos definem uma hierarquia de estruturação dos dados, enquanto atributos definem aspectos destes elementos. Assim, tipicamente, atributos serão úteis para descrever aspectos de visualização, como tamanho, altura e cor; e para localizar e identificar elementos internos e arquivos externos.

O terceiro conceito presente em um DTD são as entidades, que funcionam basicamente como macros para substituição onde for conveniente no texto, tanto do DTD quanto de arquivos XML baseados no DTD.

3.3.1.1.2 Tipos de DTDs

Um DTD pode ser armazenado internamente a um documento XML ou externamente, em um arquivo separado. No caso de DTD interno, o arquivo XML torna-se completamente auto-suficiente, podendo ser movido de um sistema para outro sem perda de suas partes, tornando mais simples seu entendimento.

Pode-se ainda utilizar DTD’s externas, que podem ser locais ou públicas, apresentando algumas vantagens importantes:

- criação de DTDs padronizadas para a comunicação de dados entre instituições, que teriam um padrão para trocar dados;
- permitem um melhor gerenciamento dos documentos, ao definir uma estrutura específica e independente dos dados da aplicação. No caso de documentos grandes, esta separação é quase obrigatória para a sua compreensão;

- facilitam a validação de documentos através de processadores específicos, descobrindo inconsistências dentro do documento.

3.3.1.2 XML Schema

XML Schema é uma alternativa de descrição de estrutura para documentos em XML que representa um avanço significativo frente a DTD, pois apresenta as seguintes vantagens:

- possui 41 tipos de dados diferentes, contra 10 na DTD;
- permite criar novos tipos de dados;
- permite a criação de uma hierarquia de objetos com herança de características;
- permite que características herdadas sejam estendidas ou restringidas;
- possui o conceito de conjuntos, onde a ordem dos elementos não é importante;
- possui o conceito de valor nulo;
- permite definir elementos substitutos, por exemplo, metrô é um substituto de trem.

3.3.1.2.1 Espaços de Nomes

Um espaço de nomes é um conjunto de tipos e elementos definidos em um arquivo em *XML Schema*. Na criação de um novo *XML Schema* estão presentes, no mínimo, dois espaços de nomes: o próprio esquema que está sendo construído e o referente à definição da linguagem *XML Schema*. Mas, dependendo da aplicação, outros espaços de nomes podem ser úteis, o que permite combinar elementos diferentes para criar um novo esquema. A cada esquema se pode associar um identificador de poucas letras, para facilitar na elaboração do documento.

3.3.1.2.2 Declarações Globais e Locais

A segunda parte de um arquivo em *XML Schema* é composta pela declaração de objetos globais. Objetos globais são aqueles visíveis por todo o esquema, sendo passíveis de serem reutilizados em uma hierarquia de generalização. Objetos locais, declarados dentro de objetos globais, são visíveis apenas dentro do seu contexto. Por objetos entende-se elemento e atributo.

3.3.1.2.3 Definição de Tipos

XML Schema possui um grande número de alternativas para a definição de tipos. Para cada tipo, um elemento XML está associado, exceto para tipos abstratos, que não possuem instâncias. Há dois formatos básicos para tipos: complexos, que possuem outros tipos ou elementos na sua composição; e simples, que descrevem a si mesmo completamente. Tipos podem, opcionalmente, possuir um nome. Tipos nomeados poderão ser reaproveitados em vários contextos. Tipos não nomeados são reconhecidos apenas pelo contexto onde estão definidos.

Tipos Simples

Tipos simples podem ser utilizados diretamente ou quando a intenção é apenas refinar um dos tipos de dados simples em *XML Schema*. Com base nos tipos simples, novos podem ser criados restringindo algumas de suas características.

Tipos Complexos

Tipos complexos são compostos por outros tipos, simples ou complexos, ou possuem atributos. A declaração de um tipo composto é realizada indicando seus componentes e o tipo de composição.

Listas

XML Schema possui um construtor para definir listas de tipos simples ou pré-existentes. Ele torna possível definir a repetição de valores de

um mesmo tipo dentro de um outro, sem caracterizar com componentes diferentes.

3.3.1.2.4 Atributos

Atributos estão associados a tipos e devem ser definidos em conjunto com o tipo. Uma restrição importante de *XML Schema* é que atributos não podem ter tipos compostos. Atributos podem, ainda, ser agrupados, permitindo a reutilização do grupo em dois tipos diferentes.

3.3.1.2.5 Derivação de Tipos

XML Schema possui uma forma restrita de herança simples de tipos, através do mecanismo de derivação. Tipos podem ser derivados por extensão, acrescentando elementos ao supertipo, ou por restrição, limitando alguma característica do supertipo, como números de ocorrências de certo elemento ou intervalos de valores aceitáveis.

3.3.1.2.6 Elementos

Elementos são a unidade básica para estruturar um arquivo em XML e não tipos. Assim, é necessário associar tipos com elementos.

3.3.1.2.7 Utilização em um Documento XML

Após definido um esquema em *XML Schema*, ele pode ser utilizado para definir a estrutura de vários documentos XML. Para tanto, no documento XML será declarado o respectivo esquema e todos espaços de nomes necessários.

3.3.1.3 XSLT

XML inclui uma linguagem que permite manipular documentos XML e transformá-los em qualquer outro tipo de arquivo texto, como HTML, RTF ou outro arquivo XML, chamada de XSLT.

Para realizar a transformação é necessário um programa auxiliar, que possa interpretar corretamente XML e XSLT e produzir um arquivo de saída com o resultado do processamento. Pode-se organizar para que ocorra no servidor, sob demanda, o que exigirá certa capacidade de processamento, ou pode ser realizado antes de disponibilizar os arquivos em um servidor, em um esquema *offline*.

Ainda, pode ocorrer que este processamento seja um componente em um sistema não relacionado à Internet.

Documentos XML são organizados em uma estrutura de árvore, com somente um elemento raiz, que pode possuir zero, um ou mais subelementos. Cada subelemento define uma nova sub-árvore, contendo zero, um ou mais subelementos, recursivamente. Ainda, elementos podem possuir atributos e conteúdo. Como seria de se esperar, comentários não são tratados por XSLT.

Arquivos com instruções em XSLT são documentos XML válidos, devendo atender às regras de sintaxe do XML. Documentos XSLT são úteis apenas para processar documentos XML. Não é possível manipular arquivos HTML, nem qualquer outro tipo de arquivo.

3.3.1.4 DOM e SAX

XSLT é uma ferramenta poderosa, mas que realiza uma tarefa específica: transformar um documento XML em outro arquivo texto. Isto não é suficiente para todos os casos, pois é necessário desenvolver ferramentas específicas para aplicações XML concretas, além de ferramentas mais genéricas para manipulação de documentos XML.

DOM e SAX são APIs independentes de linguagem e plataforma, que permitem a programas acessar e atualizar dinamicamente o conteúdo, estrutura e estilo de documentos XML. O resultado do processamento pode ser incorporado ao documento ou utilizado para outros fins, em um sistema de

informação. SAX foi definido por um conjunto de programadores, não havendo um padrão formalizado.

Uma diferença importante entre eles, é que DOM coloca todo documento XML em memória, de uma única vez, enquanto SAX trabalha componente a componente. Assim, os requisitos de memória para DOM são proporcionais ao tamanho do documento XML que for manipulado. SAX exige a criação de uma máquina de estados, o que torna a programação mais complexa.

3.3.2 SOAP

SOAP é um protocolo de mensagens extensível que forma a base para os *Web Services*, provendo um mecanismo simples e consistente que permite a uma aplicação enviar uma mensagem XML para outra aplicação.

Fundamentalmente o SOAP suporta comunicações ponto a ponto, sendo que uma mensagem SOAP é uma transmissão de uma única via de um remetente SOAP para um destinatário SOAP e qualquer aplicação pode partificar de uma transmissão atuando como um remetente ou destinatário SOAP.

As mensagens SOAP podem ser combinadas para suportar muitos tipos de comunicação, incluindo requisição/resposta, solicitação de resposta e notificação.

Em 1999 DevelopMentor, UserLand e Microsoft desenvolveram uma RPC [31] baseada em XML específica para o *Windows*. Logo depois, em 2000, Lotus e IBM juntaram-se a este esforço e ajudaram a produzir uma versão extensível e aberta da especificação que é ao mesmo tempo independente de plataforma e linguagem. Este versão da especificação, chamada de SOAP 1.1 [32] foi submetida ao W3C, o qual iniciou um esforço para a padronização.

3.3.2.1 Envelope

Um envelope SOAP fornece um mecanismo para identificar o conteúdo de uma mensagem e para explicar como a mensagem deve ser processada. O envelope inclui um *header* (cabeçalho), que provê um mecanismo extensível para fornecer directivas ou informação de controle sobre a mensagem. Por exemplo, um cabeçalho pode ser utilizado para implementar, além de outras coisas:

- transações;
- segurança, utilizando *WS-Security [33];*
- confiabilidade, com o padrão *WS-ReliableMessaging [34];*
- roteamento.

3.3.2.2 Corpo

Contém o *payload* (informação) que está sendo enviada na mensagem SOAP.

3.3.2.3 Framework de Ligações de Transporte

O SOAP 1.1 define ligações para o HTTP [35] e o *HTTP Extension Framework* [36]. Já o SOAP 1.2 define uma ligação padrão para o HTTP (não o *extension framework*) e fornece para outros ligações tais como SMTP [37], JMS [38] e outros.

3.3.2.4 Framework de Serialização

Todo o dado passado através das mensagens SOAP é codificado utilizando o XML. O dado pode ser passado como um documento XML, o qual é validado por algum documento de esboço XML (*XML Schema*) [39], ou pode ainda ser passado utilizando as regras de codificação originais do SOAP, as quais são baseadas em uma versão do *XML Schema* mais recente mas que difere dela em vários aspectos.

Existem recomendações sobre o uso da codificação SOAP em aplicativos [40]. Apesar de existir um grande número de serviços que já utilizam tal codificação eles podem ser modelados para trabalhar com o *payload* em XML puro, entretanto o procedimento mais comum é o mapeamento direto do *payload* para tipos de dados na linguagem do servidor. Este passo é conhecido como serialização e é implementado pela maioria das implementações de *Web Services*.

A serialização é possível não importando se o *payload* é um documento XML literal ou codificado em SOAP, já que ambos suportam atributos encontrados no sistema de tipos da maioria das linguagens de programação e banco de dados, incluindo tipos escalares simples, como *strings*, inteiros, *floats* e tipos complexos, tais como estruturas e *arrays*.

3.3.2.5 Estilo de Mensagens RPC

O SOAP suporta um esquema de comunicação baseado na representação RPC SOAP, a qual define uma convenção de programação que representa as requisições e respostas RPC como uma chamada de método com zero ou mais parâmetros. Quando o *payload* é construído numa estrutura simples, o elemento mais externo representa o nome do método ou operação e o mais interno os parâmetros para a operação. A resposta SOAP é similar, com o elemento mais externo de nome relacionado ao nome do método e os parâmetros mais internos representando zero ou mais parâmetros de retorno.

3.3.2.6 Estilo de Mensagens de Documento

Neste estilo, o remetente SOAP envia uma mensagem e o destinatário determina o que fazer com ela. O remetente não precisa saber absolutamente nada sobre a implementação do serviço além do formato da mensagem e o ponto de acesso URI. Fica inteiramente a cargo do destinatário determinar, baseado no conteúdo da mensagem, o que o remetende está requesitando e como processar.

Note que até com o *Document Style Messaging* é possível emular RPCs, colocando o nome do método/operação no cabeçalho *SOAPAction* da requisição HTTP.

3.3.2.7 Anexos

As ligações para anexos do SOAP (*SOAP Attachments*) [41] podem ser utilizadas para transportar dados que não sejam XML, tais como arquivos multimídia, anexados a uma mensagem SOAP. Esta especificação utiliza o MIME [42] para codificar as mensagens.

Apesar do MIME ser amplamente utilizado, ele não é o melhor mecanismo para enviar mensagens binárias em um envelope SOAP. Uma forma alternativa é o DIME [43] A especificação *WS-Attachments* [44] descreve como embutir o DIME em um envelope SOAP.

3.3.3 WSDL

WSDL é um vocabulário para a descrição de um *Web Service*. Um documento WSDL descreve quais funcionalidades ele disponibiliza, como ele se comunica e onde estã acessível, provendo um mecanismo estruturado para descrever as operações que um *Web Service* pode executar, os formatos das mensagens e os pontos de acesso de uma instância do *Web Service* [45].

As ferramentas de desenvolvimento SOAP podem utilizar uma descrição WSDL para automaticamente gerar uma interface SOAP para uso em uma aplicação.

Uma descrição WSDL define um serviço como uma coleção de pontos de rede ou portas. Cada porta é definida abstratamente como um tipo de porta, o qual suporta um conjunto de operações, cada qual processando um conjunto particular de mensagens.

Uma ligação mapeia um tipo de porta para um protocolo ou formato de dados específico e uma porta instancia um tipo de porta e o associa a um endereço de rede específico.

Uma descrição WSDL é um documento XML que contém um conjunto de definições, sendo cinco os principais elenento em um documento WSDL: tipos, mensagem, tipo de porta, ligação e serviço.

3.3.3.1 Tipos

O elemento <*types*> define os tipos de dados que são utilizados dentro das mensagens. O WSDL utiliza um sistema de tipagem de dados, o qual é um sistema de tipos canônico ainda que qualquer tipo de tipagem de dados pode ser utilizada [46].

O sistema de tipos suporta tipos escalares simples e complexos e podem ser mapeados para os sistemas de tipos da maioria das linguagens de programação e ferramentas.

As ferramentas SOAP utilizam a informação deste elemento para codificar e decodificar os dados em mensagens SOAP.

3.3.3.2 Mensagem

Um elemento <*message*> define o formato das mensagens, as quais são utilizadas como estruturas de entrada ou saída para operações. Uma mensagem pode consistir de uma ou mais partes lógicas, cada qual associada a um tipo. Quando utilizando o modelo de programação SOAP RPC cada parte representa um parâmetro de método.

3.3.3.3 Tipo de Porta

Um elemento <*portType*> define um conjunto de operações. Cada elemento <*operation*> define uma operação e as mensagens de entrada e saída associadas com a operação. Quando o modelo de programação SOAP RPC é utilizado cada operação representa um método.

3.3.3.4 Ligação

Um elemento <*binding*> mapeia as operações e mensagens definidas em um tipo de porta para uma especificação de protocolo e formato de dados completo. Por exemplo, um elemento de ligação poderia mapear um tipo de porta a uma interface SOAP RPC específica utilizando o protocolo de transporte HTTP e o sistema de codificação de dados do SOAP.

3.3.3.5 Serviço

Um elemento <*service*> define uma coleção de portas relacionadas. Este elemento mapeia uma ligação com a localização (uma URL) de uma instância de um *Web Service*.

3.3.4 UDDI

Fornece um mecanismo para registrar e categorizar *Web Services* que são disponibilizados e para localizar outros que desejam-se utilizar. Uma UDDI por si só é um *Web Service* e os usuários comunicam-se com ela através de mensagens SOAP [28].

Um registro UDDI contém informação sobre empresas e os serviços que elas oferecem. Tal informação é organizada da seguinte maneira:

- ***Business Entity:*** uma entidade de negócios contém informação sobre uma empresa incluindo seu nome, uma breve descrição e alguma informação de contato básica;
- ***Business Service:*** associado com a entidade de negócios, trata-se de uma lista de serviços oferecidos por uma entidade de negócios. Cada entrada de serviço contém uma descrição do serviço, uma lista de categorias que descrevem o serviço e uma lista de modelos de ligação que apontam para informação técnica sobre o serviço;
- ***Building Templates:*** associada a cada entrada de serviços de negócios é uma lista de modelos de ligação que provê informação sobre onde encontrar e como utilizar o serviço;

- ***Service Types:*** um tipo de serviço, definido por um montador conhecido como *tModel*, define um serviço abstrato.. Múltiplas empresas podem oferecer o mesmo tipo de serviço, todos com suporte a mesma interface de serviço. Um *tModel* especifica informação tal como seu nome, o nome da organização que o publicou, uma lista de categorias que descrevem-no e ponteiros para suas especificações técnicas.

Quando procurando por um *Web Service*, um desenvolvedor pesquisa o *UDDI Registry*, procurando por uma empresa que ofereça o tipo de serviço que ele deseja. Da entrada do *tModel* para o tipo de serviço, um desenvolvedor pode obter a descrição WSDL descrevendo a interface do serviço. Da entrada do modelo de ligação (*binding template*) para um serviço específico, o desenvolvedor pode obter a ligação ao serviço e o ponto de acesso (*access point*). Utilizando a descrição WSDL, o desenvolvedor pode construir um cliente SOAP que pode comunicar-se com o *Web Service*.

4 CLUSTERS

4.1 Introdução

Para compreender bem o tema de alta disponibilidade, é necessário entender primeiro o que são e como funcionam os *clusters* [47]. Apesar de não se depender totalmente de clusters para se montar um ambiente tolerante a falhas, recorre-se freqüentemente a eles devido à sua natureza distribuída, redundante e homogênea. Um *cluster* é um conjunto de máquinas independentes, chamadas nós, que cooperam umas com as outras para atingir um determinado objetivo comum. Por serem fracamente acopladas, para atingir este objetivo elas devem comunicar umas com as outras a fim de coordenar e organizar todas as ações a serem tomadas. Deste modo, para um usuário externo, o *cluster* é visto como sendo um único sistema. O objetivo desejado em um cluster resume-se em:

- **Alta Disponibilidade**, quando se deseja que o *cluster* forneça determinados serviços, sendo que estes devem estar sempre (ou quase sempre) disponíveis para receber solicitações. Esta probabilidade do serviço estar apto a receber solicitações é um fator dependente do cluster;
- **Alto Processamento**, quando se deseja que o *cluster* execute determinadas tarefas, sendo que elas são divididas (na sua íntegra ou em frações de uma mesma tarefa) e processadas separadamente em vários nós, a fim de que a velocidade de processamento seja incrementada.

É possível ainda ter uma situação onde o *cluster* deve atingir os dois objetivos juntos; às vezes, por razões de simplicidade, tal objetivo é atingido eliminando-se alguns rigores das definições acima.

4.2 Princípios de um Cluster

Para ser útil, um cluster precisa seguir alguns princípios básicos [47]:

- **Comodidade**: os nós em um cluster devem ser máquinas normais interconectadas por uma rede genérica. O sistema operacional também deve ser padrão, sendo que o software de gerenciamento deve ir acima dele como uma aplicação qualquer;
- **Escalabilidade**: adicionar aplicações, nós, periféricos e interconexões de rede deve ser possível sem interromper a disponibilidade dos serviços do *cluster*;
- **Transparência**: o *cluster*, que é construído com um grupo de nós independentes fracamente acoplados, deve apresentar-se como um único sistema a clientes externos ao *cluster*. Aplicações clientes interagem com o *cluster* como se fosse um único servidor com alta performance e/ou alta disponibilidade;
- **Confiabilidade**: o *cluster* deve ter capacidade de detectar falhas internas ao grupo, assim como de tomar atitudes para que estas não comprometam os serviços oferecidos;
- **Gerenciamento e Manutenção**: um dos problemas dos *clusters* é sua manutenção e configuração, que muitas vezes são tarefas complexas e propensas a gerar erros. Um mecanismo fácil de configuração e manutenção do ambiente deve existir a fim de que o *cluster* não seja um grande sistema complexo com um árduo trabalho de administração.

4.3 Clusters de Baixo Custo

4.3.1 Linux e Clusters Beowulf

Beowulf [48] nasceu com a finalidade de atender às necessidades de processamento do Earth and Space Sciences Project (ESS) no Goddard Space Flight Center (GSFC) da NASA. O objetivo inicial do projeto era obter uma estação para processamento científico (numérico) com alta capacidade e baixo custo. Na época de sua construção já haviam no mercado estações de trabalho de alto desempenho, dotadas de dois a quatro processadores, mas o uso de um maior número de subsistemas baratos em uma única estação permanecia inexplorado [49]. Há duas restrições na arquitetura da estação: ela deve usar exclusivamente hardware disponível no comércio (*commodity hardware*) e uma estação não deve custar mais do que uma estação científica de alto desempenho (em torno de US$ 50.000) [50].

4.3.2 Finalidades do Beowulf

O domínio computacional do ESS inclui o uso de grandes volumes de informação pelos cientistas, que precisam adquirir, examinar, manipular, visualizar e às vezes transformar grandes conjuntos de dados complexos, podendo envolver atividades de computação intensiva e grande movimentação de dados.

Entre as especificações do ESS constava o título "*Gigaflop Scientific Workstation*", sem que fosse claro como este objetivo seria alcançado, mas com o objetivo principal de otimizar o tempo de resposta ao usuário final. Estações de trabalho típicas não dispunham de armazenamento de massa suficiente para o volume de dados requerido, dependendo do uso de servidores de arquivos em rede. O resultado era um tempo de resposta elevado e sobrecarga de recursos compartilhados. O projeto da estação de trabalho paralela Beowulf iniciou para vencer esse desafio [50].

Embora já fosse corrente na época o uso de estações de trabalho em rede para processamento, formado o que se convencionou chamar de *cluster of workstations* (COW) ou n*etwork of workststions* (NOW), Beowulf foi criado usando o paradigma mais recente de *Pile-of-PCs* (PoPC) - pilhas de PCs – enfatizando:

- uso de componentes disponíveis no mercado de massa;
- processadores dedicados, ao invés de usar tempo ocioso de estações;
- uma rede de sistema privada.

Além disso, a estação também tem as seguintes características:

- nenhum componente feito sob encomenda;
- replicação fácil a partir de múltiplos vendedores;
- E/S escalável;
- uma base de software disponível livremente;
- uso de ferramentas de computação distribuída disponíveis livremente; com alterações mínimas;
- retorno à comunidade do projeto e melhorias.

Como vantagens desta filosofia de trabalho, os autores enumeram:

- nenhum fornecedor possui os direitos sobre o produto. Sistemas podem ser construídos usando componentes de diversas origens, graças ao uso de interfaces padrão, tais como IDE, PCI e SCSI;
- pode-se tomar vantagem das rápidas evoluções tecnológicas, permitindo adquirir sistemas mais recentes, melhores, a menores preços, capazes de continuar rodando o mesmo software. Os primeiros sistemas construídos baseavam-se no processador 80486DX4-100, enquanto os mais recentes usam Pentium Pro;

- os sistemas podem ser montados e modificados ao longo do tempo, de acordo com as necessidades e recursos (inclusive financeiros) do usuário, sem depender de configurações disponíveis de um vendedor;
- Beowulf usa software disponível livremente, tão sofisticado, robusto e eficiente quanto o comercial.

Uma característica chave do Beowulf é o uso do sistema operacional Linux [51], assim como de bibliotecas para troca de mensagens (PVM e MPI) de livre distribuição. Como mostrado mais adiante, isto permitiu fazer alterações no Linux para dotá-lo de novas características que facilitaram a implementação de aplicações paralelas.

4.3.3 Clusters SSI

Clusters Beowulf são extremamente poderosos, mas não são para todos. O primeiro empecilho dos *clusters* Beowulf é o de exigir programas especialmente projetados (escritos utilizando PVM ou MPI) para o aproveitamento dos recursos do mesmo. Isso geralmente não é um problema para aqueles que pertencem à comunidade acadêmica e de pesquisa, que estão acostumados a escrever desde o principio suas próprias aplicações de uso específico. Como eles escrevem seus códigos nos seus laboratórios, podem utilizar PVM ou MPI para criar aplicações para clusters [52].

Porém a maioria das aplicações desenvolvidas não são projetadas para utilizar PVM ou MPI, muito menos o grande público de computação tem condições de desenvolver suas próprias aplicações . Até porque muitas delas não são de uso específico. A idéia então é disponibilizar os recursos de um cluster para aplicações genéricas que não foram concebidas inicialmente para funcionar em um. Assim, aproveita-se o ganho em poder computacional que um cluster oferece para aplicações genéricas utilizadas pelo grande público de computação.

Clusters SSI surgiram para atender essa demanda. Nesse tipo de *cluster*, não existe nó mestre. A coordenação do funcionamento do *cluster* é totalmente distribuída.

Tem-se [53]:

- um domínio de *clustering*, administração, performance e balanceamento de carga;
- mesma capacidade de crescimento e sobrecarga para dois ou para dois mil nós;
- usuários não vêem os nós individualmente;
- programas não precisam ser modificados para ter proveito dos recursos (não utilizam PVM, MPI, etc.);
- configuração homogênea dos nós, balanceamento de carga automático e facilidade de gerenciamento.

4.3.3.1 MOSIX

O projeto MOSIX surgiu na Hebreu University (Israel) e é liderado pelo Prof. Ammon Barak, fundador da Mosix Inc. (também localizada em Israel). MOSIX é um acrônimo para *Multicomputer Operating System for UnIX.* O início do seu desenvolvimento se deu em 1977 e em 1979 foi completada a versão 0, chamada então de "*UNIX with Satellite Processor*s". A partir da versão 4, lançado em 1988, foi dado o nome de MOSIX.

MOSIX é um pacote de gerenciamento que permite a construção de *clusters* baseados em servidores e estações de trabalho que utilizam a arquitetura x86, que funcionam praticamente da mesma maneira que um computador com multiprocessamento simétrico [54]. Em outras palavras, implementa um cluster SSI.

O núcleo do MOSIX é composto por algoritmos adaptativos que monitoram e automaticamente respondem a requisições de recursos de todos os processos, pela disponibilidade destes no *cluster* [54].

Os algoritmos do MOSIX utilizam migração preemptiva de processos para:

- processamento paralelo, distribuindo e redistribuindo automaticamente os trabalhos;
- movimentação de processos de um nó de menor capacidade de processamento para um de maior capacidade;
- balanceamento de carga: distribuição do trabalho;
- alto desempenho de E/S: migrando processos que utilizam intensivamente E/S para um servidor de arquivos (ao contrário do tradicional modo de trazer os dados até o processo);
- E/S Paralela: migrando processos E/S paralelos de um nó cliente para um servidor de arquivos.

4.3.3.2 OpenMosix

O projeto OpenMosix surgiu a partir de uma ruptura no grupo de desenvolvimento que até então atuava no projeto MOSIX. No final do ano de 2001 as novas versões do MOSIX deixaram de ser distribuídas sob licença GPL, causando insatisfação a um grupo de desenvolvedores da Mosix Inc. Liderados então por Moshe Bar, o grupo dissidente criou o projeto OpenMosix que continuaria a ser distribuído via GPL. Além disso, Moshe fundou um empresa chamada Qlusters Inc. oferecendo soluções comerciais de *clusters* utilizando OpenMosix [55].

Este novo projeto sobrepujou o anterior em quantidade de instalações e atualmente possui quatorze colaboradores atuantes, mais que o projeto Mosix. A mudança mais crítica aconteceu nos algoritmos de migração de processos. Agora eles são baseados em modelos matemáticos de teorias econômicas, o que aumentou em muito sua eficiência.

4.3.3.3 OpenSSI

Projeto liderado por Briuce J. Walker, para implementação de um *cluster* SSI baseado no sistema operacional *Linux* [56]. Em estado inicial de desenvolvimento, muito dos colaboradores do OpenMosix também participam ou participaram deste projeto. Provavelmente será abandonado pois era patrocinado e baseado na distribuição *Linux* da Caldera International, que já não fornece mais este produto. A Hewlett Packard é também uma das patrocinadoras.

4.4 OSCAR

4.4.1 Introdução

É um apanhado dos melhores métodos conhecidos para construção, programação e uso de *clusters* [57]. Consiste em um software totalmente integrado e de fácil instalação projetado para computação de alto desempenho em *clusters*. Tudo que é preciso para instalar, configurar, manter e utilizar um “modesto” *cluster Linux* está incluído neste pacote, tornando-se desnecessária qualquer *download* ou mesmo instalação de algum pacote de software individual para o uso do *cluster*.

4.4.2 Ferramentas

4.4.2.1 Cluster Command & Control (C3)

Para simplificar sua administração, alguns desenvolvedores de clusters fazem uso do NFS [58] para gerenciar um "grande sistema de arquivos" do *cluster*; o que simplifica drasticamente o gerenciamento das configurações e desenvolvimento de aplicações, mas, ao mesmo tempo, provê um modelo menos escalável e de menor desempenho para computação em *clusters*, devido à limitações de máquina e rede.

O C3 foi projetado para descentralizar cada aspecto da configuração do cluster para ganhar em escalabilidade. Isto significa que cada nodo guarda seu próprio sistema operacional, utilizando o espaço em disco restante para aplicações temporárias e armazenamento de dados [59]. Esta abordagem gerou a necessidade de ferramentas para abstrair o fato de que um cluster consiste de inúmeras máquinas independentes. Utilizando-se de algumas ferramentas livremente disponíveis: rsync [60], OpenSSL [61], OpenSSH [62], DHCP [63] e Systemimager [64] foi desenvolvido um conjunto de ferramentas coletivamente chamadas de *Cluster Command and Control (C3) Suite*. Tais ferramentas foram projetadas para mover informação facilmente para dentro e fora do *cluster* como se ele fosse uma única máquina ao invés de um grupo de máquinas. Além disto, estas ferramentas podem ser usadas para invocar qualquer comando, em paralelo, através do *cluster*.

Os principais critérios utilizados para o projeto desta ferramenta foram:

- abstrair a presença de várias máquinas em comandos atribuídos a uma única máquina;
- promover a segurança do cluster através de autenticação de usuários externos de uma maneira segura;
- possibilitar uma maior escalabilidade, com a criação de "subclusters", também conhecidos como *clusters-of-clusters*.

4.4.2.2 LAM/MPI

4.4.2.2.1 Message Passing Interface

Um conjunto multilateral de usuários de computação paralela, empresas e pesquisadores especificaram uma interface completa para comunicação interprocesso baseada em mensagens. O padrão MPI significa portabilidade real para programas paralelos. Surgiu da necessidade de se resolver alguns problemas relacionados às plataformas de portabilidade, como restrições em relação à real portabilidade de programas, devido ao grande

número de plataformas, e o mau aproveitamento de características de algumas arquiteturas paralelas [65].

MPI fornece uma relação que permite a comunicação entre os processos de um programa paralelo com um outro processo, porém não especifica como os processos são criados, nem como ele estabelece uma comunicação. Além disso, uma aplicação MPI é estática, isto é, nenhum processo pode ser adicionado ou suprimido de uma aplicação depois de iniciada. As razões para adicionar um processo ao MPI são técnicas e práticas. As classes importantes de mensagens que passam pelas aplicações requerem um controle do processo. Estes incluem tarefas, aplicações de série com módulos paralelos e problemas que requerem uma avaliação *run-time* do número e do tipo de processos que devem ser inicializados.

Um programa de MPI consiste em processos autônomos, executando seu próprio código, em um estilo MIMD. Os códigos executados por cada processo não necessitam ser idênticos. Tipicamente, cada um é executado em seu próprio espaço de endereço, embora sejam possíveis as execuções que compartilham memória de MPI. A menos que indicado de outra maneira na especificação do padrão, MPI não coloca nenhuma exigência no resultado de sua interação com mecanismos externos que fornecem a funcionalidade similar ou equivalente.

4.4.2.2.2 Local Area Multicomputer

É um ambiente de programação e sistema de desenvolvimento em MPI para computadores heterogêneos em uma rede. Com o LAM, um *cluster* dedicado ou uma infraestrutura computacional em rede pode atuar como um computador paralelo para resolver um problema [65].

O LAM provê:

- implementação completa do padrão de comunicação MPI-1 (com a exceção de não permitir o cancelamento de *sends*);

- muitas funcionalidades do MPI-2; suporte a redes heterogêneas de estações de trabalho para incluir um cliente/servidor MPI, MPII-O, etc);
- suporte a redes de estações de trabalho heterogêneas;
- tolerância a falhas;
- comunicação rápida entre clientes.

4.4.2.3 Linux Utility for Cluster Installation

O LUI teve início como uma resposta a necessidade de um método customizado e remoto para a instalação de *clusters* para *Linux*. Ele tomou "emprestado" alguns de seus conceitos do *Network Installation Manager* (NIM) do AIX, ou seja, conceitos de objetos de máquina e recursos, gerenciamento de recursos e customização de nodos [66].

É um utilitário *open source* para a instalação de *workstations Linux* remotamente, através de uma rede *Ethernet*. O que distingue-o é que ele é "baseado em recurso". O LUI provê ferramentas para gerenciar a instalação de recursos no servidor, que podem ser alocados e aplicados para instalar clientes, permitindo aos usuários selecionar apenas quais os recursos corretos para cada cliente.

4.4.2.4 Maui PBS Scheduler

Maui é um escalonador de processos para uso em *clusters* e supercomputadores. Ele é uma ferramenta altamente otimizada e configurável, capaz de suportar uma grande quantidade de políticas de escalonamento, prioridades dinâmicas, *extensive reservations* e *fairshare*. e é reconhecido por muitos como “o mais avançado escalonador do mundo” [67]. É utilizado atualmente em centenas de órgãos governamentais, acadêmicos e comerciais ao redor do mundo. Ele aumenta a capacidade de gerência e eficiência de máquinas, desde *clusters* com alguns poucos processadores até supercomputadores.

4.4.2.5 MPICH

O projeto de disponibilizar uma implementação portável do MPI começou juntamente com o processo de definição do MPI em si. A idéia era prover um *feedback* rápido nas decisões feitas pelo *MPI Forum* e fornecer uma breve implementação para permitir aos usuários “experimentar” as definições tão logo elas fossem desenvolvidas. Os objetivos para a implementação eram incluir todos os sistemas capazes de suportar o modelo de passagem de mensagens. O MPICH é uma completa e livremente disponível implementação da especificação MPI, projetada para ser ao mesmo tempo portável e eficiente. O “CH” em MPICH quer dizer “*Chameleon*” (camaleão), o animal símbolo de adaptabilidade a um ambiente específico e por conseqüência de portabilidade. Camaleões são rápidos como o segundo objetivo do projeto, isto é, abrir mão de um pouco de eficiência em favor da portabilidade [68].

O MPICH é, assim, ao mesmo tempo, um projeto de pesquisa e um projeto de desenvolvimento de software. Como um projeto de pesquisa, seu objetivo é explorar métodos para estreitar a lacuna entre o programador de um computador paralelo e o desempenho que pode ser entregue pelo seu hardware. No MPICH recebe-se a limitação de que a interface de programação será o MPI, desprezam-se limitações na arquitetura da máquina e retém-se alto desempenho (medido em termos de banda e latência para operações de passagem de mensagem) como um objetivo. Como um projeto de software, seu objetivo é promover a adoção do padrão MPI oferecendo aos usuários uma implementação gratuita e de alto desempenho em diversas plataformas, enquanto ajudando as empresas em disponibilizar suas próprias implementações customizadas.

4.4.2.6 OpenSSH

O OpenSSH é uma versão gratuita do conjunto SSH de ferramentas de conectividade de redes que um número crescente de pessoas na Internet estão adotando. Muitos usuários de telnet [69], rlogin [70], ftp [71] e outros aplicativos similares não percebem que suas senhas são transmitidas pela

Internet em texto plano (sem serem cifradas). O OpenSSH cifra todo o tráfego (incluindo senhas) para efetivamente eliminar “escutas”, *connection hijacking* e outras formas de ataque em nível de rede [62].

4.4.2.7 Portable Batch System

O PBS é um sistema gerenciador de processos para o ambiente UNIX. Seu objetivo é atender as necessidades de usuários e sitios que precisam de um poderoso gerenciamento de trabalhos e/ou outros sistemas de gerenciamento de processos em batch. O PBS pode gerenciar um único sistema bem como *clusters* e suporta processamento desde estações de trabalho e servidores até supercomputadores [72].

Oferece [73]:

- gerenciamento da execução de trabalhos num *cluster*, utilizando filas e uma política de agendamento para melhor utilização dos recursos disponíveis;
- roteamento de trabalhos para outros sistemas de execução;
- monitoramento de trabalhos, filas e recursos de sistema.

4.4.2.8 Parallel Virtual Machine

É um subproduto de um projeto de pesquisa em andamento sobre redes de computadores heterogêneas envolvendo seus autores e suas instituições [74]. Os objetivos gerais deste projeto são investigar os problemas envolvidos e desenvolver soluções para computação concorrente heterogênea. O PVM é um conjunto de ferramentas de software integradas e bibliotecas que simulam um *framework* de computação concorrente heterogênea de uso geral e flexível para uma coleção interconectada de computadores de várias arquiteturas. O objetivo completo do sistema PVM é permitir que tal coleção de computadores sejam usados cooperativamente para computação paralela ou concorrente.

4.4.2.9 System Installation Suite

O *System Installation Suite* (SIS) foi projetado com a idéia de que um grande número de aplicações iriam interfacear com os vários aspectos do DIS para criar uma aplicação robusta, flexível e capaz de não só apenas instalar e configurar uma variedade de estações de trabalho [75]. Os objetivos a longo prazo incluem a disponibilização de uma solução para os problemas relacionados com a manutenção, atualização e reestabelecimento de redes inteiras de estações de trabalho e servidores.

Os objetivos de curto prazo e atuais, e mais realísticos, são a instalação e configuração inicial de grupos de estações de trabalho *Linux*, indo desde server farms e laboratórios de computadores até *clusters* de alta capacidade computacional.

Ele é extremamente modular e foi projetado para dar suporte a qualquer número de aplicações executadas sobre ele.

4.5 Balanceamento de Carga

4.5.1 Introdução

Por mais otimizado que seja um servidor *Web*, em *sites* de alta carga, um único *host* pode não conseguir preencher os requisitos de tolerância a falhas, escalabilidade e principalmente desempenho, demandados por estes sites.

O consumo de recursos para o atendimento de múltiplas conexões simultâneas, como memória e poder de processamento, pode facilmente atingir o limite do hardware.

Upgrade de hardware, em geral, é uma solução paliativa e que não tem uma boa relação custo/benefício. Além disso, a causa de latência pode não ser apenas o servidor em si, mas a largura de banda, o que demanda a localização diferenciada dos servidores [76]. A arquitetura baseada em

múltiplos servidores é uma alternativa efetiva e relativamente barata que tem sido adotada pela maioria dos grandes *sites*.

Um mecanismo de distribuição de carga ideal deve prover a melhor associação entre clientes e servidores, objetivando o melhor tempo de resposta, de modo mais transparente possível ao usuário.

Pode-se classificar técnicas de distribuição de carga em dois grupos: Local e Global. O primeiro se concentra na distribuição de carga sobre servidores distribuídos localmente, fisicamente próximos, sendo útil quando se atinge o limite de hardware de um servidor. Já o segundo se foca na utilização de servidores dispersos geograficamente e é adotado quando o problema de desempenho se concentra no limite da rede.

Atualmente a segunda técnica ainda é pouco explorada, com poucas soluções apresentadas se comparada com a primeira, sendo assim abordada em menor profundidade neste trabalho.

4.5.2 Benefícios

O balanceamento de carga pode ser definido como o processo de distribuição de requisições entre um grupo de servidores e tem-se tornado um elemento importante nas redes pois provê um aumento da escalabilidade, alto desempenho, alta disponibilidade e melhoria na recuperação após tragédias [77].

- **Escalabilidade:** o balanceamento de carga faz com que múltiplos servidores pareçam um único servidor (um único serviço virtual), distribuindo transparentemente as requisições dos usuários entre eles. Isto permite que servidores adicionais possam ser adicionados rapida e transparentemente aos usuários finais - uma flexibilidade extremamente útil para grandes corporações e provedores de serviços;
- **Alto Desempenho:** o melhor desempenho é conseguido quando o poder de processamento dos servidores é utilizado inteligentemente. Boas soluções de balanceamento de carga redirecionam requisições a serviços para os

servidores que estão menos carregados e, portanto, capazes de responder mais rapidamente. Necessariamente, um dispositivo de balanceamento de carga deve ser capaz de gerenciar o tráfego agregado de múltiplos servidores. Se tal dispositivo tornar-se um "gargalo" ele deixa se ser uma solução para tornar-se um problema adicional;

- **Disponibilidade:** se uma aplicação ou servidor falha o balanceamento de carga pode automaticamente redistribuir as requisições para outros servidores, além de evitar que interrupções planejadas para manutenção de software ou hardware afetem a disponibilidade de um serviço;
- **Recuperação após Tragédias:** as requisições podem ser redirecionadas para instalação de *backup* quando alguma falha catastrófica desabilita a estrutura principal.

4.5.3 Algoritmos

- ***Round robin*:** um algoritmo simples que distribui cada nova conexão/sessão para o próximo servidor disponível;
- ***Round robin* balanceado:** uma melhoria do método round robin onde os tempos de resposta de cada servidor dentro de um serviço virtual são constantemente medidos para determinar qual servidor será responsável pela próxima conexão/sessão.
- **Menos Conexões:** determina o servidor que receberá a próxima conexão mantendo um registro de quantas conexões cada servidor está provendo. O servidor com o menor número de conexões receberá a próxima requisição.

O algoritmo de *round robin* pode ser eficaz para a distribuição de carga entre servidores com igual capacidade de processamento. Quando os servidores possuem capacidades diferentes, a utilização dos tempos de resposta ou número de conexões ativas como critério de seleção pode otimizar os tempos de resposta.

4.5.4 Servidores Distribuídos Localmente

A distribuição local de servidores, em um único ponto da Internet é útil para a situação onde um único *host* não está sendo capaz de atender a demanda de requisições, embora a rede suporte a carga.

Dado um conjunto de servidores que hospedam um site em uma única localização, pode-se identificar duas arquitetura principais:

- **Arquitetura baseada em *cluster***: onde os nós servidores ocultam seus endereços IP na visão dos clientes. O único endereço IP visível é o do servidor virtual correspondente a um dispositivo (hardware ou software) localizado entre os clientes e o conjunto de servidores, responsável pela distribuição de carga, denominado distribuidor.;
- **Arquitetura Web Distribuída**: quando os endereços IP reais dos nós servidores são visíveis às aplicações.

A arquitetura distribuída é mais antiga, onde o roteamento é, em geral, decidido pelo sistema DNS [78]. A solução baseada em cluster é mais recente e o roteamento de requisições é inteiramente conduzido pelos componentes internos do cluster, o que proporciona um maior controle na atribuição de tarefas aos servidores, além de mecanismos de segurança e alta disponibilidade. Em geral, a arquitetura distribuída é mais aconselhável para servidores dispersos geograficamente.

4.5.4.1 Arquitetura Baseada em Cluster

Um *Web cluster* corresponde a um conjunto de servidores que são posicionados em uma localização única, inter-conectados por uma rede de alta velocidade e que apresentam uma única imagem (endereço IP). Cada servidor do cluster em geral apresenta seu próprio disco e um sistema operacional completo.

Embora o *cluster* possa ser composto por dezenas de nós, somente um nome e um endereço IP é divulgado. Assim, o servidor DNS autoritativo

executa o mapeamento de um para um, traduzindo o nome no endereço IP de um entidade dedicada a rotear requisições entre os servidores, fazendo com que a distribuição de carga seja totalmente transparente aos clientes. Desta forma, esta entidade atua como um distribuidor centralizado que possui um controle bastante fino sobre o escalonamento a ser feito.

O distribuidor pode ser implementado tanto como um dispositivo de hardware dedicado plugado na rede, ou como módulos de software de um sistema operacional de propósito geral ou não. Em geral, a primeira alternativa é a de melhor desempenho, porém menos flexível e mais cara.

4.5.4.1.1 Roteamento

O distribuidor pode identificar univocamente cada servidor real através de um endereço que pode estar em diferentes níveis do protocolo, dependendo da arquitetura utilizada. Mais especificamente, um servidor do cluster pode ser identificado através do próprio endereço IP ou ainda do endereço MAC, da camada de enlace.

As informações que o distribuidor utiliza para decisões de roteamento também se baseiam nos níveis de rede, e servem para classificar o tipo de distribuidor. A escolha do mecanismo de roteamento a se utilizar possui um grande impacto nos algoritmos de escalonamento, uma vez que as informações disponíveis para decisão são bastante distintas. Atualmente existem dois tipos de distribuidor: o de nível 4 e o de nível 7 (referente às camadas do Modelo ISO/OSI).

4.5.4.1.2 Nível 4

Executa o chamado roteamento "cego" em relação ao conteúdo da camada de aplicação, uma vez que a escolha do servidor do cluster é feita no momento do estabelecimento da conexão TCP, na chegada do primeiro SYN. Visto que os pacotes do cliente não “sobem” até a camada de aplicação, a distribuição é mais rápida. Todavia, as políticas de escalonamento não podem se basear no conteúdo das requisições do cliente.

Pelo fato de pacotes pertencentes à mesma conexão TCP deverem ser associados ao mesmo servidor real, o distribuidor mantém uma tabela de associação, relacionando cada sessão TCP com o servidor adequado. Em geral a tabela é mantida em memória e acessada através de uma função de *hash*, para a melhoria de desempenho.

Isso permite que requisições HTTP, sob a mesma conexão TCP, sejam atendidas por um único servidor, de forma a evitar o overhead associado com estabelecimento e fechamento de conexões TCP. Todavia, isso não garante o conceito de transação no nível de aplicação, uma vez que diferentes requisições HTTP sob uma mesma transação no nível de aplicação podem ser direcionados para servidores distintos.

4.5.4.1.3 Nível 7

Pode executar o roteamento baseado no conteúdo. Primeiramente o distribuidor estabelece uma conexão TCP completa com o cliente, examina a requisição HTTP no nível de aplicação (nível 7) e repassa a conexão para um servidor escolhido. É menos eficiente em relação ao distribuidor de nível 4 porque possui um mecanismo de roteamento com informações mais detalhadas, além da necessidade de cópia de dados e mudanças de contexto entre modo *kernel* e usuário. Por outro lado, pode suportar políticas de roteamento mais sofisticadas.

Por trabalhar no nível 7, pode implementar em si mesmo o conceito de transação no nível de aplicação, usando informações de estado como *cookies* para direcionar requisições ao servidor adequado. Isso é útil em aplicações como o comércio eletrônico ou para melhoria de desempenho em aplicações que exigem a transmissão de dados via SSL [79] (onde a fase de estabelecimento de transação é bastante custosa), dentre outras aplicações.

Outra vantagem desta arquitetura é a possibilidade de se combinar o cacheamento das páginas estáticas no distribuidor, porque é possível verificar qual arquivo está sendo requisitado no momento do roteamento. Esta solução é conhecida como proxy reverso [80].

4.5.4.2 Arquitetura Web Distribuída

Como visto, nesta arquitetura cada servidor *Web* não oculta seu endereço IP real em relação aos clientes e não existe a figura do distribuidor fisicamente entre os clientes e os servidores. Os mecanismos de roteamento aplicados nesta arquitetura podem ser classificados de acordo com o local onde é tomada a decisão sobre qual servidor atenderá a requisição do cliente. Este local pode ser no próprio cliente, no serviço de resolução de nomes ou nos servidores.

4.5.4.2.1 Distribuição Selecionada pelo Cliente

A forma mais básica de se efetuar a distribuição de carga é se fornecer uma lista de servidores com suas respectivas localizações geográficas, deixando a cargo do usuário a seleção de qual servidor utilizar.

Embora seja bastante simples de se implementar, já que não se adiciona nenhuma complexidade ao servidor, é bastante deselegante, por não ser transparente ao usuário, e não se basear em nenhum critério de seleção, senão a posição geográfica. Nem sempre o servidor mais próximo é o que provê melhor tempo de resposta. Além disso, para o caso de múltiplos servidores localizados em um mesmo cluster (não dispersos geograficamente), não haverá nenhum critério de escolha para o usuário, a não ser o “psicológico”, e em geral os primeiros da lista sofrerão maior carga. Não é possível se ter nenhum controle da distribuição de carga pelos administradores.

4.5.4.2.2 Mecanismo de roteamento por DNS

O DNS foi a primeira solução proposta para se distribuir a carga entre múltiplos servidores *Web* e continua sendo uma das mais simples. Originalmente foi concebida para múltiplos servidores distribuídos localmente. Todavia, atualmente é mais utilizado para servidores *Web* geograficamente distribuídos.

O DNS demanda a presença de múltiplas cópias do site em servidores idênticos.

Este mecanismo funciona da seguinte forma: para um nome de domínio são associados múltiplos endereços IP correspondentes a servidores distintos. Quando é feita uma consulta a um domínio, uma lista de servidores é retornada, variando-se a ordenação desta lista a cada consulta. Em geral o navegador utiliza como endereço IP o primeiro da lista. Como para cada requisição DNS o primeiro da lista varia, clientes distintos acessam endereços IPs distintos correspondentes a diferentes servidores. Isto provê uma forma básica de distribuição de carga, baseada no algoritmo de round-robin.

Embora bastante fácil de se aplicar, este mecanismo possui várias desvantagens:

- a distribuição de carga é completamente “cega”, ou seja, não se utiliza de nenhum critério, como carga atual, congestionamento ou proximidade geográfica. Isso obriga que todos os servidores tenham a mesma capacidade de processamento, e estejam localizados geograficamente próximos. Assim, pode-se considerar que o Round Robin DNS puro não é um mecanismo de balanceamento de carga completo [81];
- dificulta a implementação de aplicações que dependem da manutenção do estado, ou transação como as de comércio eletrônico. Resultados DNS são armazenados em cache pelo cliente durante um tempo chamado TTL. Caso a transação seja longa, de maior duração que o TTL, é possível que o cliente continue a transação a partir de um outro servidor. Para o tratamento coerente desse tipo de requisição é necessária a utilização de um *timeout* de transação ou então que o servidores compartilhem dados sobre o estado do usuário. Ambas as alternativas adicionam complexidade. *Round Robin DNS* é mais indicado quando o conteúdo é somente para leitura;
- a natureza distribuída do DNS dificulta o controle dos endereços IP a serem utilizados, visto que a propagação dos dados não é instantânea, e consultas podem ficar armazenadas em cache de servidores DNS intermediários. Em suma, os *caches* intermediários acabam ignorando a distribuição de carga

provida pelo DNS. Assim, no caso da retirada de um dos servidores do cluster este ainda pode continuar sendo requisitado por resoluções DNS armazenadas em cache de clientes ou servidores DNS intermediários, dentro do período TTL. Uma solução para isso seria a divulgação do registro DNS com TTL igual a zero, o que obriga uma nova consulta ao servidor "*authoritative*" a cada requisição, o que traz como efeito colateral, o aumento do tráfego DNS. Mesmo assim, não há detecção automática de falha em algum dos servidores, ou seja, não provê tolerância a falhas.

Embora o DNS não seja um mecanismo de distribuição de carga ideal, pelas deficiências apresentadas, ele continua sendo o mais utilizado pela sua facilidade de implementação e baixo custo. Além disso, não demanda o posicionamento dos servidores em uma única sub-rede, como as soluções baseadas em *cluster* discutidas anteriormente.

Existem propostas de mudanças no *Round-Robin DNS* com algoritmos de roteamento mais sofisticados [82], além de uma outra modificação ao DNS para melhor distribuição de carga [83].

5 ESTUDO DE CASO

Inspirados pela grande utilização de PDAs para a realização das atividades comerciais das grandes distribuidoras de produtos, tais como as de bens alimentícios, decidiu-se implementar um sistema distribuído a ser utilizado pelos vendedores de tais empresas. O sistema permite a verificação em tempo real dos preços e disponibilidade em estoque dos produtos, assim como a realização de pedidos de compra.

A seguir é dada uma explanação sobre os aspectos relevantes a escolha do tipo de serviço a ser implementado neste estudo de caso.

5.1 O Sistema

O estudo de caso implementa um sistema que tem como função a automação do processo de vendas para distribuidoras de produtos em atacado. Hoje, tais empresas já se encontram em estado avançado de informatização e já utilizam aplicativos para emissão de pedidos que funcionam em PDAs. Porém os vendedores necessitam se deslocar até a sede para realizar a transferência dos pedidos armazenados em seus PDAs para o sistemas de gerenciamento interno das distribuidoras. Algumas investiram em dispositivos que permitem conexão aos sistemas via telefone, através da utilização de modems. Porém ambas as soluções possuem um problema: o longo espaço de tempo decorrido entre o registro do pedido de compra de um cliente pelo vendedor, e a efetiva notificação da distribuidora de que um pedido foi realizado.

Num mercado cada vez mais competitivo, o pronto atendimento às solicitações dos clientes e uma boa logística para gerenciamento de estoques são fundamentais para o sucesso comercial de uma distribuidora. Satisfação dos clientes e redução de custos nas operações são os objetivos a serem alcançados.

Uma ferramenta que permitisse o envio de pedidos e a recuperação de informações sobre produtos em tempo real, permitiriam as distribuidoras otimizar os seus processos, planejar melhor seu estoque e terem um melhor relacionamento com seus clientes.

5.1.1 Sistema Operacional

Cada PDA possui um sistema operacional responsável por permitir o uso do mesmo. São raras as empresas que desenvolvem o seu próprio SO. Todas as que foram citadas anteriormente, exceto a Apple, licenciam um sistema operacional de terceiros. E esse mercado é disputado principalmente Microsoft [13] e pela PalmSource [84]. A Microsoft licencia duas versões do Windows para PDAs: o Windows CE e o recentemente Windows Micro Edition. Já a PalmSource licencia o PalmOS.

O sistema operacional escolhido para a realização do estudo de caso foi o PalmOS 5, em detrimento aos da Microsoft. A principal razão é o desejo de trabalhar numa plataforma com características bem diferentes das que se utiliza para o desenvolvimento de aplicações para computadores pessoais. Implementar programas para a plataforma Windows da Microsoft é extremamente semelhante para as suas diferentes versões: tanto para PDAs quanto para computadores pessoais. Portanto, para obter um superior ganho em conhecimento e experiência, optou-se pela solução da PalmSource.

5.1.2 Ferramentas

5.1.2.1 Cliente

O aplicativo de vendas para o PDA foi implementado utilizando-se o kit de desenvolvimento de software disponibilizado pela PalmSource, para o sistema operacional Palm OS 5. No entanto, os programas desenvolvidos com esse kit funcionam normalmente em versões anteriores. O mesmo é composto por um conjunto de arquivos de cabeçalho e bibliotecas. Não é fornecido nenhum outro aplicativo.

Para então poder criar os programas, o pacote de aplicativos PRC-Tools foi usado. O mesmo é composto por versões modificadas dos compiladores da GNU das linguagens C e C++ para plataforma Palm, ferramentas para depuração, criação de arquivos de recursos, emulador do Palm, entre outros. Tal pacote é destinado a ser utilizado em computadores rodando o sistema operacional GNU/Linux, embora existe um porte para o ambiente Cygwin. Devido a forte integração entre o PRC-Tools e o GNU/Linux, o último foi escolhido para desenvolvimento do sistema.

Inicialmente foi definido que utilizaria-se C++ como a linguagem de implementação, mas as dificuldades em conciliar os conceitos de programação orientada ao objeto ao kit de desenvolvimento da PalmSource inviabilizaram essa opção. Portanto acabou-se por programar o sistema em linguagem C.

Por fim, para ter poder acessar o *Web Service* utilizamos o *Web Service Toolkit* para PalmOS desenvolvido em parceiria pela PalmSource e IBM. Essa biblioteca é baseada em outra, de código aberto, a gSOAP.

5.1.2.2 Servidor

O servidor SOAP a ser utilizado pelo aplicativo cliente foi implementado em PHP [85], utilizando-se de bibliotecas do PEAR [86] (SOAP e banco de dados). Tal servidor roda sobre um servidor Apache [87],

acessando um banco de dados MySQL [88] e, no caso deste estudo, todos sendo executados no mesmo servidor, sob o sistema operacional Linux.

Vale a pena lembrar que o sistema foi primeiramente desenvolvido com as mesmas características em um servidor FreeBSD [89]. Isto significa que ele pode ser executado (hipoteticamente) em qualquer sistema operacional onde possa ser instalado um servidor Apache, com suporte a PHP e estando instaladas as bibliotecas necessárias.

O banco de dados não necessariamente precisa estar sendo executado no mesmo servidor do sistema. Além disso, poder-se-ia ter sido utilizado qualquer um dos bancos de dados relacionais suportados pela biblioteca de banco de dados do PEAR (PEAR::DB), ou seja, DBase, FastBaseSQL, Interbase, Informix, MSQL, MS-SQL, MySQL, MySQLi, Oracle 8, ODBC, PostgreSQL, SQLLite e Sybase. Apenas simples e rápidas alterações seriam necessárias para a troca de banco de dados.

Tais escolhas quanto aos servidores, ferramentas e bibliotecas utilizadas foram feitas com o intuito de proporcionar uma grande flexibilidade, escalabilidade, alto desempenho e tolerância a falhas do sistema. já que o processamento do sistema (e do banco de dados) pode ser distribuído em um cluster, os nodos do cluster podem estar executando praticamente qualquer sistema operacional e banco de dados, ficando a critério do administrador do sistema a melhor estratégia a ser utilizada, de acordo com sua necessidade.

5.1.3 Análise e Projeto

Para a construção de todo o conjunto de programas do estudo de caso, utilizou-se um processo de desenvolvimento de software adaptado do *Unified Process* (UP), baseando-se na descrição do mesmo feita por Larman [90]. Como o estudo de caso implementa apenas um conjunto relativamente reduzido de operações, todo o projeto foi elaborado em apenas uma iteração.

Na etapa de análise, foram fabricados um sumário executivo, listas de requisitos funcionais e não funcionais e por fim um caso de uso. Um

simples diagrama de conceitos identifica os principais elementos observados na análise de sistema. Toda a documentação gerada está no Anexo A deste documento.

Como não se tinha experiência anterior de desenvolvimento de Web Services nem de aplicações para a plataforma Palm OS, na etapa de projeto apenas alguns artefatos foram produzidos. Não foi realizado um planejamento minucioso de como os sistemas cliente e servidor seriam implementados. Não foi possível avaliar que abordagens para implementação ou estruturação dos sistemas seria mais adequada.

Apenas se definiu que operações seriam disponibilizadas via o protocolo SOAP pelo servidor a serem usadas pelo cliente rodando no Palm, além da definição da estrutura do banco de dados. Um simples contrato descreve as operações de sistema disponibilizadas pelo Web Service, um diagrama entidade-relacionamento demonstra a estrutura do banco de dados e um diagrama de classes descreve a estrutura global da implementação dos serviços no servidor. Todos disponibilizados também no Anexo A.

5.1.4 Implementação

O código fonte do sistemas servidor e cliente estão documentados no Anexo B. Como dito anteriormente, utilizou-se a linguagem de programação PHP para implementação do *Web Service* e para o cliente na plataforma Palm OS usou-se a linguagem de programação C.

6 CONCLUSÃO

6.1 Dificuldades Encontradas

Foram diversas as dificuldades e/ou obstáculos encontrados durante o trabalho. A maioria dos obstáculos foram superados, mas tais problemas limitaram a a constatação da viabilidade das idéias expostas.

Primeiramente pode-se citar a ausência de um ambiente no Departamento de Informática e Estatística para que o sistema fosse “colocado a prova” com características semelhantes às que seriam encontradas em um ambiente de produção, sendo acessado por um número consideravelmente grande de clientes e onde problemas inerentes a tal tipo de situação pudessem ser detectados e solucionados.

Após definido como o sistema seria estruturado, o pequeno número de publicações nesta área tornou mais difícil a tarefa de escolher quais ferramentas utilizar para a implementação (que linguagens e quais bibliotecas seriam as mais adequadas). Além disso, visto que o trabalho trata de um tema ainda pouco explorado, as ferramentas selecionadas para a implementação do sistema, tanto na parte do servidor e principalmente no cliente para Palm OS, mostraram-se um pouco “imaturas”, já que elas estão em suas primeiras versões.

Por fim, o curto espaço de tempo disponível para a pesquisa, modelagem e implementação do sistema dificultou a execução do estudo de caso. Como o tema do trabalho foi alterado apenas cerca dois meses antes de

sua apresentação, o tempo para desenvolvimento tornou-se reduzido e isto exigiu uma dedicação extra para que o sistema estivesse operacional em tempo hábil.

6.1.1 Cliente

Com o estudo de caso constatou-se que a tarefa de construção de sistemas que funcionem na plataforma Palm OS exige um grande esforço de programação. Um exemplo: para exibir a lista de clientes na tela e permitir o usuário selecionar um deles foi necessário escrever código para desenhar cada um dos nomes dos clientes, navegar entre a listagem e colocar em destaque o cliente selecionado.

Uma limitação da plataforna Palm OS torna complica mais ainda o trabalho do programador: a instrução jmp dos processadores da família Motorola 68000 podem apenas pular diretamente para extremos de segmento de código de tamanho menor ou igual a 32 kB, limitando o tamanho do programa executável. Como os *stubs* (componentes de programa responsáveis pelo tratamento do serviço SOAP) gerados pelo *Web Service Toolkit* em nosso estudo de caso são maiores que 32 kB quando compilados, faz-se necessário dividir o programa em vários segmentos de código.

O compilador utilizado permite dividir o código fonte do programa em várias seções, e cada seção é então convertida para um segmento de código. Com isso tem-se uma a aplicação contendo vários segmentos de código. As chamadas de função ou procedimento a partir de um segmento para outro são especialmente tratadas [91]. Porém realizar a divisão do programa fonte em seções é uma tarefa manual e suscetível a erros. Não se consegue prever onde deverá ser criada novas seções. As novas seções podem ainda quando compiladas serem maior que 32 kB, exigindo ainda uma subdivisão. Por fim, se houver algum erro na identificação de procedimentos e funções que pertencem a um seção específica, o compilador não detecta o erro e o sistema irá se comportar de maneira imprevisível.

6.1.2 Servidor

Existe uma certa dificuldade para manipulação de tipos complexos do SOAP, utilizando-se a biblioteca PEAR::SOAP. O pouco tempo de desenvolvimento impediu que a biblioteca fosse alterada, afim de melhorar a interface para a manipulação deste tipo de dado. Então evitou-se utilizar tipos complexos.

Apesar da biblioteca PEAR::SOAP ainda estar em desenvolvimento (beta), esta apresenta a maioria das funcionalidades necessárias para implementar-se um servidor SOAP.

O servidor, da maneira que foi modelado, pode beneficiar-se enormemente do poder de processamento de um *cluster*, apesar deste ponto não ter sido muito explorado no estudo de caso.

Percebeu-se também uma facilidade na adição de novos serviços ao servidor. Por um lado, ele gera dinamicamente um documento WSDL. Assim, a medida que novos serviços forem adicionados, o WSDL do servidor estará sempre atualizado, possibilitando que os clientes se adequem aos novos serviçõs e/ou alterações nos já existentes. Além disso, para que um novo serviço seja adicionado basta que sejam geradas as classes necessárias e que o novo serviço seja adicionado à lista de serviços do servidor.

6.2 Considerações Finais

Por serem protótipos, o mínimo de funcionalidade é apresentada no sistemas implementados neste estudo de caso. Não foram realizados testes suficientes para avaliar a vialidade de seu uso comercialmente em razão da falta de estrutura, já comentada no início deste capítulo.

6.3 Trabalhos Futuros

6.3.1 Cliente

Dado o grande esforço de implementação do cliente para a plataforma Palm OS, como um possível trabalho futuro é a utilização de J2ME (versão de Java para dispositivos portáteis) como plataforma de desenvolvimento. Assim pode-se até realizar um comparativo entre as vantagens e desvantagens de cada uma das opções.

Outra possibilidade é a implementação de SOAP alternativo ao *Web Service Toolkit*. Tendo-se o cuidado de automatizar a tarefa de divisão em seções do código, assim contornando a limitação de tamanho da área de código do aplicativo (visto anteriormente).

Por fim, dado também as dificuldades enfrentadas durante o desenvolvimento do cliente, torna-se imprescindível o desenvolvimento um *framework* para construção de interfaces gráficas de aplicações para o Palm OS. Isso reduziria o esforço despendido.

6.3.2 Servidor

Uma possibilidade visualizada durante este estudo de caso seria a utilização da estrutura de transporte de mensagens SMS nas operadoras como meio de transporte das mensagens (envelopes SOAP) entre cliente e servidor.

Além disso poderiam ser feitas baterias de testes no sistema, simulando-se o acesso de um número considerãvel de clientes, acessos simultâneos, etc. Tais testes serviriam para analisar o comportamento do sistema em um ambiente de produção.

Brevemente estará disponível um módulo SOAP para o PHP, assim como já existem módulos para XML, XML-RPC, etc. Isto possibilitará uma execução mais eficiente das funções relacionadas a *Web Services*. Um estudo

de caso poderia ser feito utilizando-se tais funções afim de verificar o impacto real no desempenho do sistema fazendo uso desta abordagem.

Outras alternativas de balanceamento de carga também poderiam ser analisadas afim de encontrar-se a melhor solução para este tipo de sistema.

Neste estudo de caso, o potencial de processamento paralelo do *cluster* não foi totalmente explorado. Uma possibilidade para o melhor uso de tal poder de processamento seria a utilização de soluções de bancos de dados que tenham seu processamento distribuído entre os diversos nós do *cluster*.

ANEXO A – ARTEFATOS DE ANÁLISE E PROJETO

Sumário Executivo

O vendedor, utilizando-se do seu Smartphone, poderá a partir do sistema selecionar um cliente para atendimento, buscar informações sobre preço e estoque de produtos em tempo real, e realizar pedidos de compra para o cliente, que serão imediatamente enviados para a distribuidora.

Para tanto, terá-se-á um serviço acessível através do sistema sendo executado no Palm disponibiliza operações básicas como a coleta de informações sobre produtos (preço e estoque), clientes e permita a realização de pedidos de compra.

Requisitos do Sistemas

Neste trabalho, denomina-se requisitos funcionais aqueles que definem uma funcionalidade no sistema. Já os requisitos não funcionais, são as propriedades que o sistema deverá ter. Construiu-se Listas de Requisitos a partir dos modelos de Larman [90].

Requisitos Funcionais

Sistema Aplicativo de Vendas para o SmartPhone.

.Nº.	***Requsito***
RF1	Exibir lista de clientes no início da execução do sistema

.Nº.	***Requsito***
RF2	Permitir a seleção de um cliente.
RF3	Prover busca de mais informações sobre o cliente selecionado, via *Web Service*.
RF4	Escolher o cliente para o qual será realizada as consultas informações de possíveis pedidos.
RF5	Listar produtos.
RF6	Permitir a seleção de um produto.
RF7	Permitir busca de dados sobre preço e estoque de um produto via *Web Service*.
RF8	Armazenar uma lista de pedidos de clientes.
RF9	Adicionar um produto selecionado a lista de pedidos, informando a sua quantidade.
RF10	Visualizar a lista de pedidos
RF11	Remover pedido da lista.
RF12	Armazenar a lista de pedidos de maneira persistente.
RF13	Prover mecanismo para cancelar e apagar a lista de pedidos.
RF14	Enviar a lista de pedidos dos clientes para o *Web Service*.
RF15	Atualizar lista de clientes via *Web Service*
RF16	Atualizar lista de produtos via *Web Service*

Requisitos Não Funcionais

Descrição: Sistema aplicativo de vendas para o *Smartphone*.

Nº	***Requisito***
RN1	Ter interface gráfica consistente, segundo as recomendações da PalmSource.
RN2	Funcionar em SmatPhones com 2 MB de memória.
RN3	Poder operar o sistema mesmo sem ter acesso ao *Web Service*
RN4	Implementado para funcionar em *smartphones* que utilizam o sistema operacional PalmOS.

Requisitos Funcionais

Descrição: *Web Service.*

.Nº.	***Requsito***
RF1	Prover serviço que oferece informações sobre clientes.
RF2	Prover serviços que oferece informações sobre preço e estoque de produtos
RF3	Prover serviço que retorne uma listagem com informações reduzidas sobre produtos.
RF4	Prover serviço que retorne uma listagem com informações reduzidas sobre clientes.
RF5	Prover serviço para a efetivação de pedidos de compra.
RF6	Cadastrar Produtos.
RF7	Cadastrar Clientes.
RF8	Alterar informações de Produtos.
RF9	Alterar informações de Clientes.
RF10	Na recepção de um pedido de compra, já dar baixa no estoque.
RF11	Armazenar de maneira persistente os pedidos de compra.
RF12	Armazenar de maneira persistente os dados de produtos.
RF13	Armazenar de maneira persistente os dados de clientes.

Requisitos Não Funcionais

Descrição: *Web Service.*

Nº	***Requisito***
RN1	Os serviços deverão ter tempo de resposta inferior a 1 segundo.
RN2	Deve ter capacidade para lidar com alta carga de utilização dos serviços.
RN3	Ter garantia de Alta Disponibilidade.
RN4	Serviços implementados via protocolo SOAP para troca de mensagems.

Caso de Uso: Realizar pedido de Compra

Atores:

Vendedor, Cliente

Finalidade:

Fazer pedido de compra de produtos para o cliente.

Visão Geral:

O vendedor vai até o cliente verificar se o mesmo necessita de algum produto. O cliente lhe informa quais deseja comprar e o vendedor realiza o pedido via o sistema funcionando em seu *Smartphone*.

Seqüência Típica de Eventos:

1. O vendedor chega ao estabelecimento comercial cliente da distribuidora para verificar a necessidade de aquisição de produtos, selecionando o mesmo no sistema.
2. O vendedor solicita ao cliente um produto que necessita adquirir,
3. O cliente informa o produto e sua quantidade.
4. O vendedor seleciona o produto e verifica as informações de preço e estoque, via *Web Service*.
5. O cliente analisa as informações de preço e estoque e confirma o pedido.
6. O Vendedor registra o pedido de compra no sistema.
7. O Vendedor pergunta ao cliente se existem mais produtos a serem adquiridos.
8. Se o cliente responder que sim, voltar para o passo 2. Continuar em caso contrário.
9. O vendedor lista todos os pedidos de compra registrados no sistema e mostra ao cliente, esperando sua confirmação.

10.O cliente confirma a lista de pedidos de compra,

11.O Vendedor solicita ao sistema para enviar a lista de compras via *Web Service*.

12.O pedido de compras é realizado.

Seqüências Alternativas:

5a. O cliente não confirma o pedido de compra de um produto. Pular para o passo 7.

10a. O cliente não confirma a lista de pedidos de compra. Cancelar caso de uso.

Diagrama de Casos de Uso

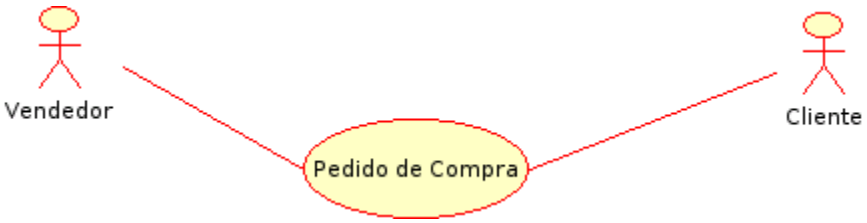
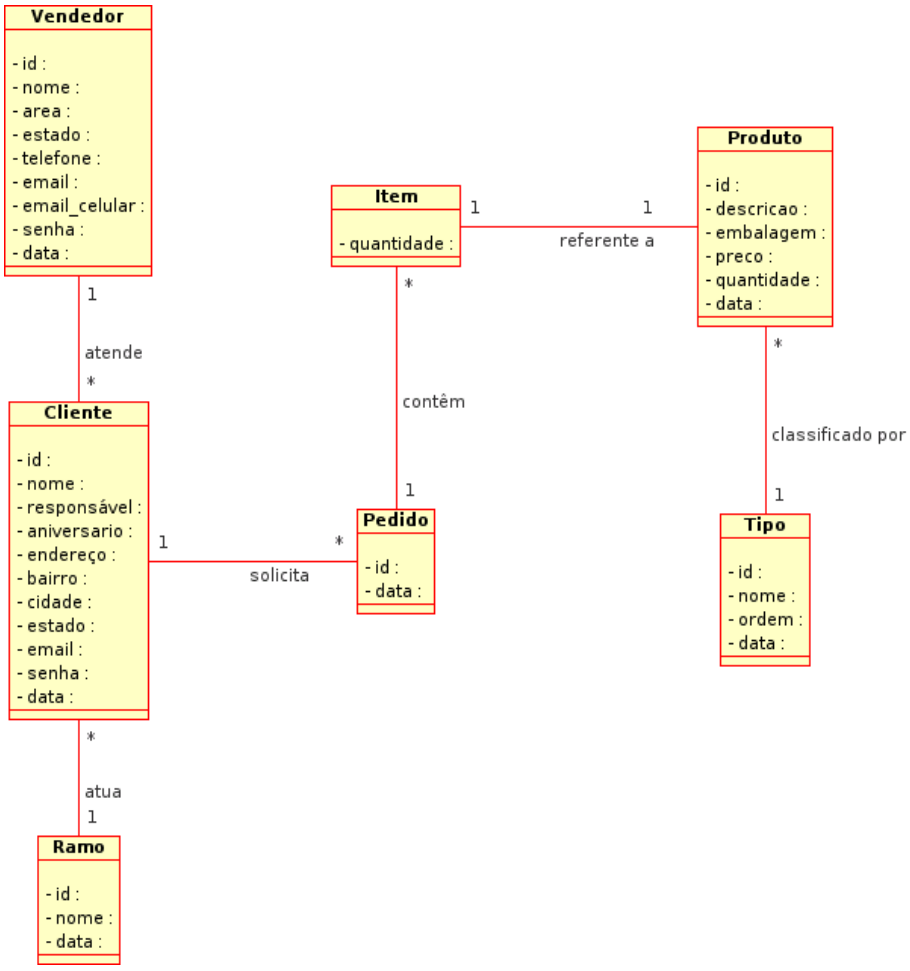


Diagrama de Conceitos



Arquitetura do Sistema

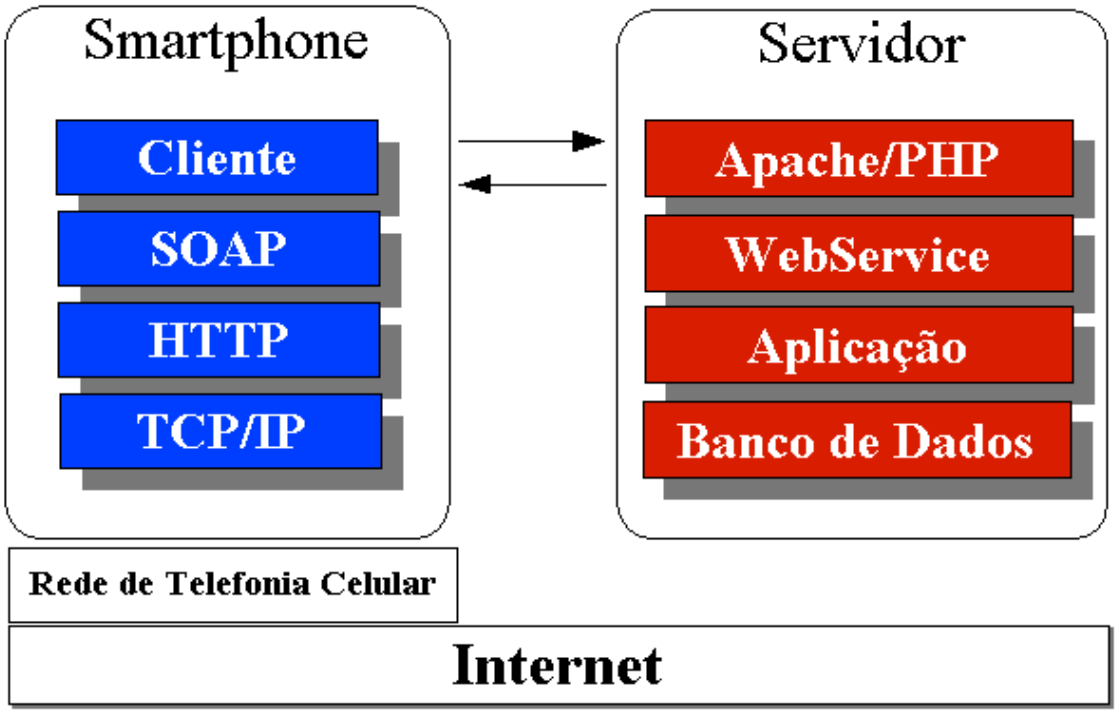
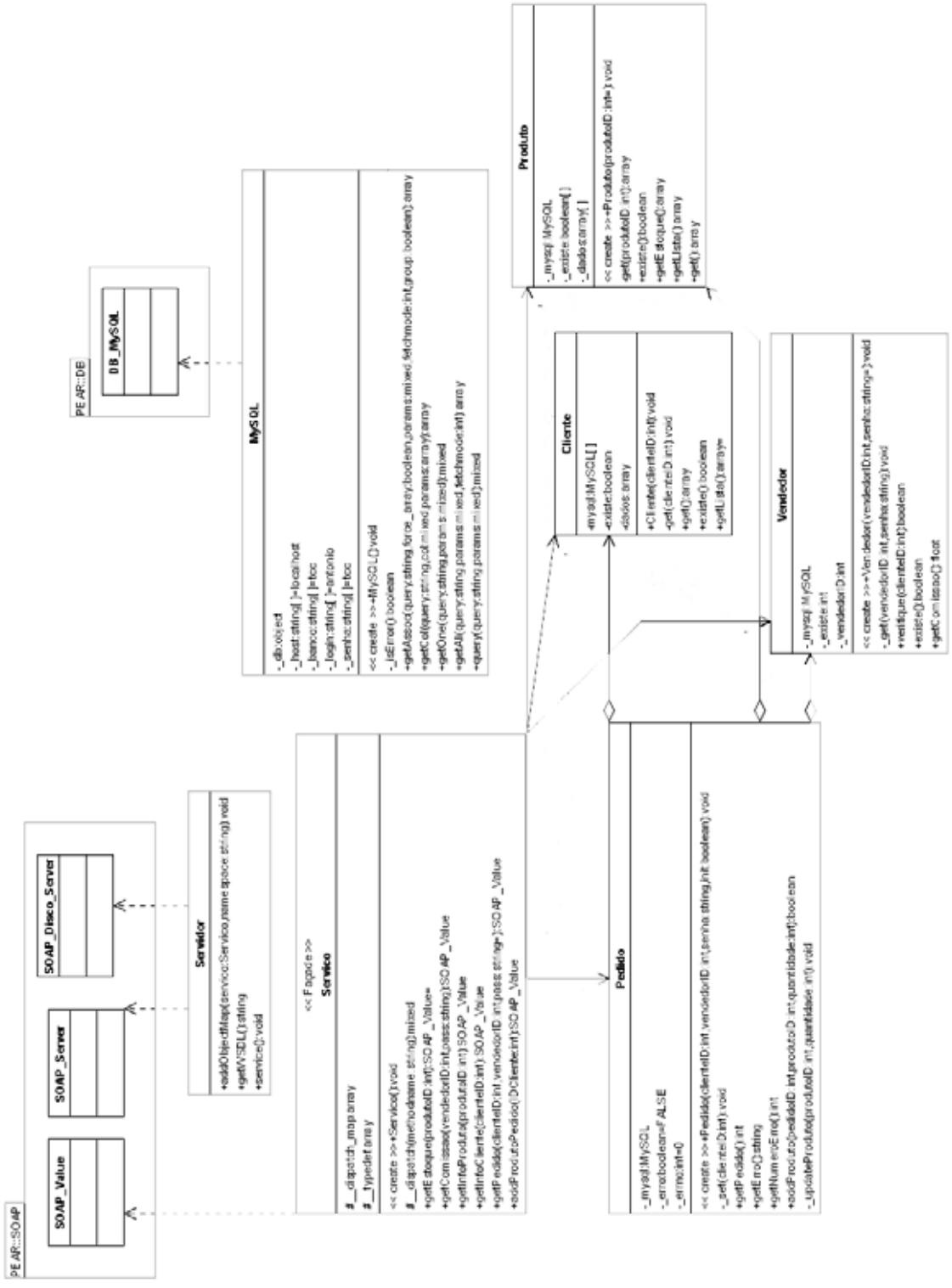
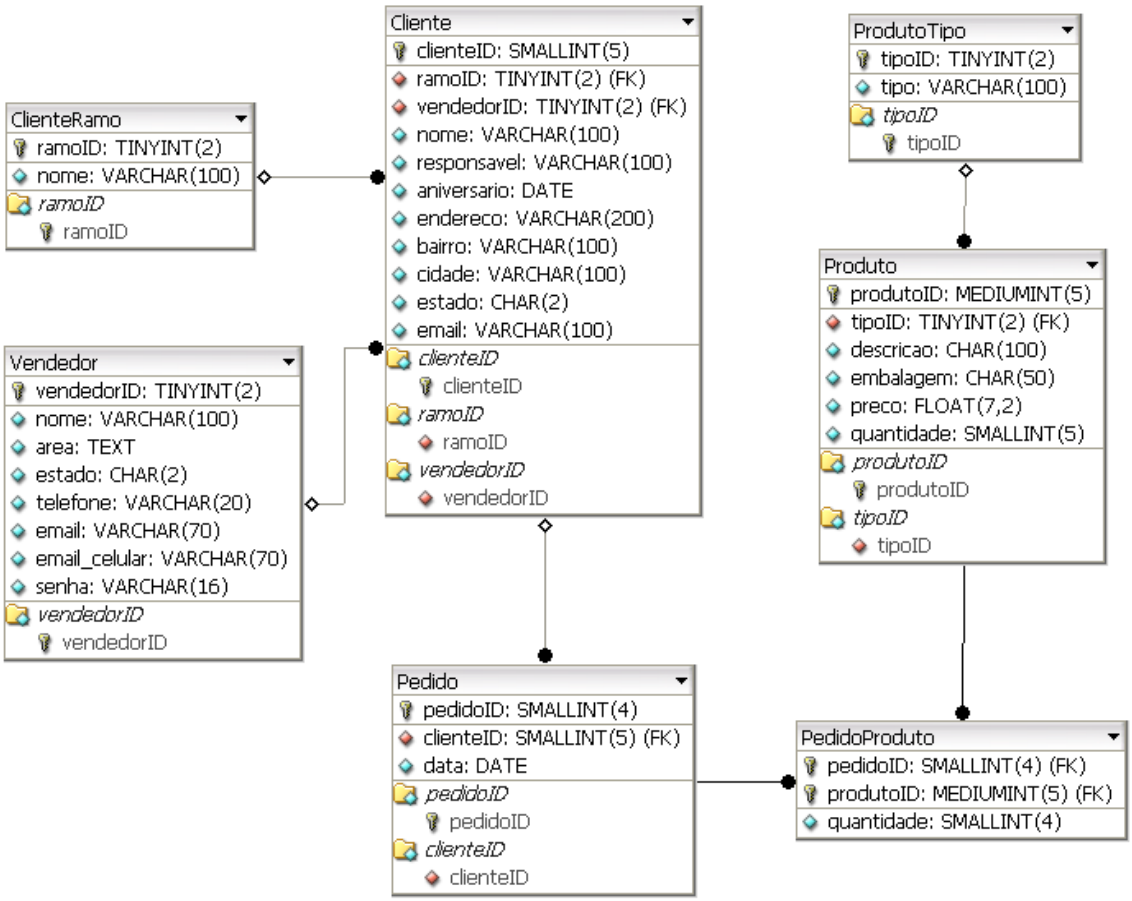


Diagrama de Classes do Servidor



Modelagem do Banco de Dados



ANEXO B – CÓDIGO FONTE DOS SISTEMAS

Cliente Palm

selcliente.h:

```
#ifndef _SELCLIENTE_H_
#define _SELCLIENTE_H_

#include "PalmOS.h"
#include "vendasprc.h"

Boolean SelClienteHandleEvent(EventType* event);
#endif
```

selcliente.c:

```
#include <TextMgr.h>
#include <ErrorMgr.h>
#include <SoundMgr.h>
#include <TimeMgr.h>
#include <AboutBox.h>
#include <Category.h>
#include <Menu.h>
#include <UIResources.h>
#include <StringMgr.h>

#include "SelCliente.h"
#include "vendasprc.h"
#include "List.h"
#include "VendasDbs.h"
#include "VendasTypes.h"
#


extern DmOpenRef RamoDbPtr;
extern DmOpenRef ClienteDbPtr;
extern ClienteType ClienteCorrente;


/* Coluna ou colunas da tabela */
#define NOME_CLIENTE_COLUMN 0

/* Valores utilizados para fazer rolagem acelerada */
#define SCROLL_DELAY 2
#define SCROLL_ACCELERATION 2
```

```
#define SCROLL_SPEED_LIMIT 5

/* variáveis globais do módulo*/

static UInt16 LastSecond = 0;
static UInt16 ScrollUnits = 0;
static UInt16 TopVisibleRecord = 0;
static UInt16 SelectedRow = 0;

static    char* ramo = "Supermercado";
static    char* endereco = "Beira Mar Norte";
static    char* contato = "João da Silva";


/* funções internas deste  módulo */

static void SelClienteListInit(FormPtr frmP);

static void SelClienteLoadTable(FormPtr frmP);

static UInt16 SelClienteNumberOfRows(TablePtr table);

static Boolean SelClienteSeekRecord (UInt16 * indexP, Int16 offset,
Int16 direction);

static void SelClienteUpdateScrollButtons(FormPtr frmP);

static void SelClienteDrawRecord(void * table, Int16 row, Int16
column, RectanglePtr bounds);

static void SelClienteAlteraClienteCorrente();

static void SelClientePreencheCamposInfoCliente();



Boolean SelClienteHandleEvent(EventType *event)
{
    Boolean handled = false;
    FormPtr frmP;

    switch (event->eType)
    {
        case frmOpenEvent:
        {
            frmP = FrmGetActiveForm();

            /* inicializa a listagem (table) de clientes */
            SelClienteListInit(frmP);


            FrmDrawForm (frmP);
            handled = true;
            break;
         }
        case ctlSelectEvent:
        {
            if (event->data.ctlSelect.controlID == SCFInfoBtn)
            {
                SelClienteAlteraClienteCorrente();
                frmP = FrmInitForm(InfoClienteForm);
                SelClientePreencheCamposInfoCliente(frmP);
                FrmDoDialog(frmP);
```

```
                FrmDeleteForm(frmP);
                handled = true;

            }
            else if(event->data.ctlSelect.controlID ==
SCFSelecionaBtn)
            {
                FrmGotoForm(SelProdutoForm);
            }
            else if (event->data.ctlSelect.controlID == SCFSairBtn)
            {
                EventType event;
                MemSet(&event, sizeof(event), 0);
                event.eType = appStopEvent;
                EvtAddEventToQueue(&event);
            }
            handled = true;
            break;
        }
        case tblSelectEvent:
        {
            SelectedRow = event->data.tblSelect.row;
            handled = true;
            break;
        }

        default:
        {
             break;
        }

    }
    return handled;
}



static void SelClienteListInit(FormPtr frmP)
{
    UInt16 row;
    UInt16 rowsInTable;
    TableType* tblP;
    ControlPtr ctl;

    SysFatalAlert("Inicializar tabela");


    /* pega o ponteiro da table*/
    tblP = (TableType*) GetObjectPtr(SCF_CLIENTE_TABLE);

    rowsInTable = TblGetNumberOfRows(tblP);
    /* Inicializa os estilos das células da tabela e as desabilita */
    for (row = 0; row < rowsInTable; row++)
    {
        TblSetItemStyle(tblP, row, NOME_CLIENTE_COLUMN,
customTableItem);
        TblSetRowUsable(tblP, row, false);
    }

    /* habilita o uso das colunas */
    TblSetColumnUsable(tblP, NOME_CLIENTE_COLUMN, true);

    /* Define que registros privados não deve sem exibidos.
```

```
       Sem função em nosso programa, por isso desabilitada */
    TblSetColumnMasked(tblP, NOME_CLIENTE_COLUMN, false);

    /* configura a rotina que desenha cada célula da tabela */
    TblSetCustomDrawProcedure(tblP, NOME_CLIENTE_COLUMN,
SelClienteDrawRecord);

//    SysFatalAlert("Chamando SelClienteLoadTable");

    /* Carrega os registros dentro da tabela de clientes*/
    SelClienteLoadTable(frmP);

    /* modifica o título do trigger de ramo */
    ctl = (ControlPtr) GetObjectPtr(SCF_RAMO_POPUP);
}

static void SelClienteLoadTable(FormPtr frmP)
{
    UInt16 row;
    UInt16 numRows;
    UInt16 lineHeight;
    UInt16 recordNum;
    UInt16 visibleRows;
    FontID currFont;
    TableType* tblP;
    UInt16 attr;
    Boolean masked;

    /* pega ponteiro da tabela */
    tblP = (TableType*) GetObjectPtr(SCF_CLIENTE_TABLE);

    /* desativa frescuras no item selecionado */
    TblUnhighlightSelection(tblP);

    /* calcula quantas linhas devem ser exibidas, assim como
       o índice do primeiro registro a ser exibido.  */
    visibleRows = SelClienteNumberOfRows(tblP);
    recordNum = TopVisibleRecord;
    /* tenta encontrar o último registro a ser exibido */
    if (!SelClienteSeekRecord(&recordNum, visibleRows -1,
dmSeekForward))
    {
        /* Tem-se pelo menos uma linha na table que não exibirá um
registro.
           Arrumar a mesma para que isso não aconteça */
        TopVisibleRecord = dmMaxRecordIndex;
        if (!SelClienteSeekRecord(&TopVisibleRecord, visibleRows-1,
dmSeekBackward))
        {
            /* sem registros suficientes para preencher uma página ...
*/
            TopVisibleRecord = 0;
            SelClienteSeekRecord(&TopVisibleRecord, 0, dmSeekForward);
        }
    }

    /* pega o tamanho de uma linha com a fonte utilizada para
       desenhar as células */
    currFont = FntSetFont(stdFont);
    lineHeight = FntLineHeight();
    FntSetFont(currFont);

    /* iterage por todas as linhas associando associando a cada uma
```

```
       o ID do registro que deve ser exibido na mesma, e outras coisas
mais */
    numRows = TblGetNumberOfRows(tblP);
    recordNum = TopVisibleRecord;

    for (row = 0; row < visibleRows; row++) {
        if (!SelClienteSeekRecord(&recordNum, 0, dmSeekForward))
break;

        /* marca a linha como utilizável */
        TblSetRowUsable(tblP, row, true);

        /* marca a linha como inválida, para a mesma ser redesenhada
*/
        TblMarkRowInvalid(tblP, row);

        /* armazena o ID do registro como o ID da linha */
        TblSetRowID(tblP, row, recordNum);

        /* define a fonte e a altura da linha */
        TblSetItemFont(tblP, row, NOME_CLIENTE_COLUMN, stdFont);
        TblSetRowHeight(tblP, row, lineHeight);

        recordNum++;
    }

    /* esconde as linhas que não tem nenhum dado */
    while (row < numRows)
    {
        TblSetRowUsable(tblP, row, false);
        row++;
    }

    /* desenha a tabela */
    SelClienteUpdateScrollButtons(frmP);

}
```

```
/**
 * Retorna o número de linhas da table que
 * devem ser exibidas na tela.
 */
static UInt16 SelClienteNumberOfRows(TablePtr table)
{
    UInt16 rows;
    UInt16 rowsInTable;
    UInt16 tableHeight;
    FontID currFont;
    RectangleType r;

    /* pega o número de linhas da tabela */
    rowsInTable = TblGetNumberOfRows(table);

    /* pega as coordenadas das bordas da tabela */
    TblGetBounds(table, &r);
    tableHeight = r.extent.y;
```

```
    currFont = FntSetFont(stdFont);
    rows = tableHeight / FntLineHeight();
    FntSetFont(currFont);

    if (rows <= rowsInTable)  return (rows); else return
(rowsInTable);

}


/**
 * Procuta o próximo registro existente de um determinado ramo
 * a partir de um offser, retornando se encontrou ou não e define
 * o valor de indexP como o índice do mesmo.
 */
static Boolean SelClienteSeekRecord (UInt16 * indexP, Int16 offset,
Int16 direction)
{
    UInt16 numRecords;
    DmSeekRecordInCategory(ClienteDbPtr, indexP, offset, direction,
dmAllCategories );

    if (DmGetLastErr()) return (false);
      return (true);
}


static void SelClienteDrawRecord(void * table, Int16 row, Int16
column, RectanglePtr bounds)
{
    UInt16 recordNum;
    Err error;
    ClienteRecord *cliente;
    MemHandle recordHandle;
    MemPtr recordPtr;
    FontID currFont;
    UInt16 maxWidthPixels;
    Boolean truncated;
    UInt16 maxWidth;

    recordNum = TblGetRowID(table, row);
    recordHandle = DmGetRecord(ClienteDbPtr, row);
    if (recordHandle == NULL) {
        SysFatalAlert("Registro não encontrado");
        return;
    }
    recordPtr = MemHandleLock(recordHandle);
    if (recordPtr==NULL) {
        SysFatalAlert("Não foi possível travar área de memória");
        return;
    }
    cliente = (ClienteRecord*) recordPtr;

    currFont = FntSetFont(stdFont);

    maxWidth = StrLen(cliente->nome);
    maxWidthPixels = bounds->extent.x;
    FntCharsInWidth(cliente->nome, &maxWidthPixels, &maxWidth,
&truncated);
    WinDrawChars(cliente->nome, maxWidth, bounds->topLeft.x, bounds-
>topLeft.y);

    FntSetFont(currFont);
```

```
    MemHandleUnlock(recordHandle);
    DmReleaseRecord(ClienteDbPtr, recordNum, false);
}


/**
 *  Função que mostra ou esconde os scroll buttons.
 */
static void SelClienteUpdateScrollButtons(FormPtr frmP)
{
    UInt16 row;
    UInt16 upIndex;
    UInt16 downIndex;
    UInt16 recordNum;
    Boolean scrollableUp;
    Boolean scrollableDown;
    TableType *tblP;

    /* Verifica se existem registros anteriores ao primeiro da table,
       e então habilita ou desabilita o scrollUp */
    recordNum = TopVisibleRecord;
    scrollableUp = SelClienteSeekRecord(&recordNum, 1,
dmSeekBackward);

    /* pega registro na última linha da tabela */
    tblP = (TableType*)  GetObjectPtr(SCF_CLIENTE_TABLE);
    row = TblGetLastUsableRow(tblP);
    if (row != tblUnusableRow)
        recordNum = TblGetRowID(tblP, row);

    /* Verifica se tem mais registros além do último exibido,
habilitando
       ou não o scrollDown */
    scrollableDown = SelClienteSeekRecord(&recordNum, 1,
dmSeekForward);

    /* atualiza os botões de scroll */
    /*
    upIndex = FrmGetObjectIndex(frmP, ListUpButton);
    downIndex = FrmGetObjectIndex(frmP, ListDownButton);
    FrmUpdateScrollers(frmP, upIndex, downIndex, scrollableUp,
scrollableDown);
    */
}

static void SelClienteAlteraClienteCorrente()
{
    Err erro;
    UInt16 recordId;
    TableType *tblP;
    MemHandle record;
    MemPtr recordPtr;
    ClienteRecord* cliente;
    UInt16 stringSize;

    tblP = (TableType*) GetObjectPtr(SCF_CLIENTE_TABLE);

    if (SelectedRow == 0) {
        ErrNonFatalDisplayIf(true, "Selecione um cliente");
    }

    recordId = TblGetRowID(tblP, SelectedRow);
```

```
    record  = DmGetRecord(ClienteDbPtr, recordId);
    recordPtr = MemHandleLock(record);
    cliente = (ClienteRecord*) recordPtr;

    ClienteCorrente.clienteId = cliente->codigo;
    ClienteCorrente.ramoId = cliente->ramo;
    StrNCopy(ClienteCorrente.nome, cliente->nome, CLIENTE_NOME_SIZE);

    MemHandleUnlock(record);
    DmReleaseRecord(ClienteDbPtr, recordId, false);
}



static void SelClientePreencheCamposInfoCliente(FormPtr frmP)
{
    FieldPtr field;
    MemHandle codigoString;
    MemPtr ptr;

    codigoString = MemHandleNew(maxStrIToALen);
    ptr = MemHandleLock(codigoString);
    StrIToA(ptr, ClienteCorrente.clienteId);
    MemHandleUnlock(codigoString);
    field = GetObjectPtrForm(frmP, ICF_CODIGO_FLD);
    FldSetText(field, codigoString,0 ,maxStrIToALen);
    field = GetObjectPtrForm(frmP, ICF_NOME_FLD);
    FldSetTextPtr(field, ClienteCorrente.nome);

    field = GetObjectPtrForm(frmP, ICF_RAMO_FLD);
    FldSetTextPtr(field, ramo);

    field = GetObjectPtrForm(frmP, ICF_ENDERECO_FLD);
    FldSetTextPtr(field, endereco);

    field = GetObjectPtrForm(frmP, ICF_CONTATO_FLD);
    FldSetTextPtr(field, contato);

}
```

selproduto.c:

```
#include "SelProduto.h"
#include "VendasTypes.h"
#include "VendasDbs.h"
#include "vendasprc.h"


/* Coluna ou colunas da tabela */
#define NOME_CLIENTE_COLUMN 0

/* Valores utilizados para fazer rolagem acelerada */
#define SCROLL_DELAY 2
#define SCROLL_ACCELERATION 2
#define SCROLL_SPEED_LIMIT 5

/* variáveis globais do módulo*/

static UInt16 LastSecond = 0;
static UInt16 ScrollUnits = 0;
```

```
static UInt16 TopVisibleRecord = 0;
static UInt16 SelectedRow = 0;

static ProdutoType ProdutoCorrente;

extern DmOpenRef RamoDbPtr;
extern DmOpenRef ClienteDbPtr;
extern DmOpenRef ProdutoDbPtr;

extern ClienteType ClienteCorrente;

static char* preco = "40.00";
static char* estoque = "100";



static void SelProdutoListInit(FormPtr frmP);

static void SelProdutoLoadTable(FormPtr frmP);

static UInt16 SelProdutoNumberOfRows(TablePtr table);

static Boolean SelProdutoSeekRecord (UInt16 * indexP, Int16 offset,
Int16 direction);

static void SelProdutoUpdateScrollButtons(FormPtr frmP);

static void SelProdutoDrawRecord(void * table, Int16 row, Int16
column, RectanglePtr bounds);

static void SelProdutoAlteraProdutoCorrente();

static void SelProdutoPreencheCamposInfoCliente();


Boolean SelProdutoHandleEvent(EventType* event)
{

    Boolean handled = false;
    FormPtr frmP;

    switch (event->eType)
    {
        case frmOpenEvent:
            frmP = FrmGetActiveForm();
            SelProdutoListInit(frmP);
            FrmDrawForm (frmP);
            handled = true;
            break;
        case ctlSelectEvent:
            if (event->data.ctlSelect.controlID == SPFInfoBtn)
            {
                SelProdutoAlteraProdutoCorrente();
                frmP = FrmInitForm(InfoProdutoForm);
                SelProdutoPreencheCamposInfoProduto(frmP);
                FrmDoDialog(frmP);
                FrmDeleteForm(frmP);
                handled = true;

            }
            break;
        default:
            break;
```

```
    }
    return handled;

}



static void SelProdutoListInit(FormPtr frmP)
{
    UInt16 row;
    UInt16 rowsInTable;
    TableType* tblP;
    ControlPtr ctl;

    SysFatalAlert("Inicializar tabela");


    /* pega o ponteiro da table*/
    tblP = (TableType*) GetObjectPtr(SPF_PRODUTO_TABLE);

    rowsInTable = TblGetNumberOfRows(tblP);
    /* Inicializa os estilos das células da tabela e as desabilita */
    for (row = 0; row < rowsInTable; row++)
    {
        TblSetItemStyle(tblP, row, NOME_CLIENTE_COLUMN,
customTableItem);
        TblSetRowUsable(tblP, row, false);
    }

    /* habilita o uso das colunas */
    TblSetColumnUsable(tblP, NOME_CLIENTE_COLUMN, true);

    /* Define que registros privados não deve sem exibidos.
       Sem função em nosso programa, por isso desabilitada */
    TblSetColumnMasked(tblP, NOME_CLIENTE_COLUMN, false);

    /* configura a rotina que desenha cada célula da tabela */
    TblSetCustomDrawProcedure(tblP, NOME_CLIENTE_COLUMN,
SelProdutoDrawRecord);

//    SysFatalAlert("Chamando SelProdutoLoadTable");

    /* Carrega os registros dentro da tabela de clientes*/
    SelProdutoLoadTable(frmP);

    /* modifica o título do trigger de ramo */
    //ctl = (ControlPtr) GetObjectPtr(SCF_RAMO_POPUP);
}

static void SelProdutoLoadTable(FormPtr frmP)
{
    UInt16 row;
    UInt16 numRows;
    UInt16 lineHeight;
    UInt16 recordNum;
    UInt16 visibleRows;
    FontID currFont;
    TableType* tblP;
    UInt16 attr;
    Boolean masked;
```

```
    /* pega ponteiro da tabela */
    tblP = (TableType*) GetObjectPtr(SPF_PRODUTO_TABLE);

    /* desativa frescuras no item selecionado */
    TblUnhighlightSelection(tblP);

    /* calcula quantas linhas devem ser exibidas, assim como
       o índice do primeiro registro a ser exibido.  */
    visibleRows = SelProdutoNumberOfRows(tblP);
    recordNum = TopVisibleRecord;
    /* tenta encontrar o último registro a ser exibido */
    if (!SelProdutoSeekRecord(&recordNum, visibleRows -1,
dmSeekForward))
    {
        /* Tem-se pelo menos uma linha na table que não exibirá um
registro.
           Arrumar a mesma para que isso não aconteça */
        TopVisibleRecord = dmMaxRecordIndex;
        if (!SelProdutoSeekRecord(&TopVisibleRecord, visibleRows-1,
dmSeekBackward))
        {
            /* sem registros suficientes para preencher uma página ...
*/
            TopVisibleRecord = 0;
            SelProdutoSeekRecord(&TopVisibleRecord, 0, dmSeekForward);
        }
    }

    /* pega o tamanho de uma linha com a fonte utilizada para
       desenhar as células */
    currFont = FntSetFont(stdFont);
    lineHeight = FntLineHeight();
    FntSetFont(currFont);

    /* iterage por todas as linhas associando associando a cada uma
       o ID do registro que deve ser exibido na mesma, e outras coisas
mais */
    numRows = TblGetNumberOfRows(tblP);
    recordNum = TopVisibleRecord;

    for (row = 0; row < visibleRows; row++) {
        if (!SelProdutoSeekRecord(&recordNum, 0, dmSeekForward))
break;

        /* marca a linha como utilizável */
        TblSetRowUsable(tblP, row, true);

        /* marca a linha como inválida, para a mesma ser redesenhada
*/
        TblMarkRowInvalid(tblP, row);

        /* armazena o ID do registro como o ID da linha */
        TblSetRowID(tblP, row, recordNum);

        /* define a fonte e a altura da linha */
        TblSetItemFont(tblP, row, NOME_CLIENTE_COLUMN, stdFont);
        TblSetRowHeight(tblP, row, lineHeight);

        recordNum++;
    }

    /* esconde as linhas que não tem nenhum dado */
    while (row < numRows)
```

```
    {
        TblSetRowUsable(tblP, row, false);
        row++;
    }

    /* desenha a tabela */
    SelProdutoUpdateScrollButtons(frmP);

}




/**
 * Retorna o número de linhas da table que
 * devem ser exibidas na tela.
 */
static UInt16 SelProdutoNumberOfRows(TablePtr table)
{
    UInt16 rows;
    UInt16 rowsInTable;
    UInt16 tableHeight;
    FontID currFont;
    RectangleType r;

    /* pega o número de linhas da tabela */
    rowsInTable = TblGetNumberOfRows(table);

    /* pega as coordenadas das bordas da tabela */
    TblGetBounds(table, &r);
    tableHeight = r.extent.y;

    currFont = FntSetFont(stdFont);
    rows = tableHeight / FntLineHeight();
    FntSetFont(currFont);

    if (rows <= rowsInTable)  return (rows); else return
(rowsInTable);

}


/**
 * Procuta o próximo registro existente de um determinado ramo
 * a partir de um offser, retornando se encontrou ou não e define
 * o valor de indexP como o índice do mesmo.
 */
static Boolean SelProdutoSeekRecord (UInt16 * indexP, Int16 offset,
Int16 direction)
{
    UInt16 numRecords;
    DmSeekRecordInCategory(ProdutoDbPtr, indexP, offset, direction,
dmAllCategories );

    if (DmGetLastErr()) return (false);
      return (true);
}
```

```
static void SelProdutoDrawRecord(void * table, Int16 row, Int16
column, RectanglePtr bounds)
{
    UInt16 recordNum;
    Err error;
    ProdutoRecord *produto;
    MemHandle recordHandle;
    MemPtr recordPtr;
    FontID currFont;
    UInt16 maxWidthPixels;
    Boolean truncated;
    UInt16 maxWidth;

    recordNum = TblGetRowID(table, row);
    recordHandle = DmGetRecord(ProdutoDbPtr, row);
    if (recordHandle == NULL) {
        SysFatalAlert("Registro não encontrado");
        return;
    }
    recordPtr = MemHandleLock(recordHandle);
    if (recordPtr==NULL) {
        SysFatalAlert("Não foi possível travar área de memória");
        return;
    }
    produto = (ProdutoRecord*) recordPtr;

    currFont = FntSetFont(stdFont);

    maxWidth = StrLen(produto->descricao);
    maxWidthPixels = bounds->extent.x;
    FntCharsInWidth(produto->descricao, &maxWidthPixels, &maxWidth,
&truncated);
    WinDrawChars(produto->descricao, maxWidth, bounds->topLeft.x,
bounds->topLeft.y);

    FntSetFont(currFont);

    MemHandleUnlock(recordHandle);
    DmReleaseRecord(ProdutoDbPtr, recordNum, false);
}


/**
 *  Função que mostra ou esconde os scroll buttons.
 */
static void SelProdutoUpdateScrollButtons(FormPtr frmP)
{
    UInt16 row;
    UInt16 upIndex;
    UInt16 downIndex;
    UInt16 recordNum;
    Boolean scrollableUp;
    Boolean scrollableDown;
    TableType *tblP;

    /* Verifica se existem registros anteriores ao primeiro da table,
       e então habilita ou desabilita o scrollUp */
    recordNum = TopVisibleRecord;
    scrollableUp = SelProdutoSeekRecord(&recordNum, 1,
dmSeekBackward);

    /* pega registro na última linha da tabela */
    tblP = (TableType*)  GetObjectPtr(SPF_PRODUTO_TABLE);
```

```
    row = TblGetLastUsableRow(tblP);
    if (row != tblUnusableRow)
        recordNum = TblGetRowID(tblP, row);

    /* Verifica se tem mais registros além do último exibido,
habilitando
       ou não o scrollDown */
    scrollableDown = SelProdutoSeekRecord(&recordNum, 1,
dmSeekForward);

    /* atualiza os botões de scroll */
    /*
    upIndex = FrmGetObjectIndex(frmP, ListUpButton);
    downIndex = FrmGetObjectIndex(frmP, ListDownButton);
    FrmUpdateScrollers(frmP, upIndex, downIndex, scrollableUp,
scrollableDown);
    */
}

static void SelProdutoAlteraProdutoCorrente()
{
    Err erro;
    UInt16 recordId;
    TableType *tblP;
    MemHandle record;
    MemPtr recordPtr;
    ProdutoRecord* produto;
    UInt16 stringSize;

    char* embalagem = "Engradado de 24 garrafas";


    tblP = (TableType*) GetObjectPtr(SPF_PRODUTO_TABLE);

    if (SelectedRow == 0) {
        ErrNonFatalDisplayIf(true, "Selecione um produto");
    }

    recordId = TblGetRowID(tblP, SelectedRow);

    record  = DmGetRecord(ProdutoDbPtr, recordId);
    recordPtr = MemHandleLock(record);
    produto = (ProdutoRecord*) recordPtr;

    ProdutoCorrente.produtoId = produto->codigo;
    ProdutoCorrente.tipoId = produto->tipo;
    StrNCopy(ProdutoCorrente.descricao, produto->descricao,
PRODUTO_DESCRICAO_SIZE);
    StrNCopy(ProdutoCorrente.embalagem, embalagem,
PRODUTO_EMBALAGEM_SIZE);
    ProdutoCorrente.preco = 40.0;
    ProdutoCorrente.estoque = 100;

    MemHandleUnlock(record);
    DmReleaseRecord(ProdutoDbPtr, recordId, false);
}



static void SelProdutoPreencheCamposInfoProduto(FormPtr frmP)
{
    FieldPtr field;
```

```
    MemHandle precoString;
    MemHandle estoqueString;
    MemPtr ptr;


    field = GetObjectPtrForm(frmP, IPFDescricaoFld);
    FldSetTextPtr(field, ProdutoCorrente.descricao);

    field = GetObjectPtrForm(frmP, IPFEmbalagemFld);
    FldSetTextPtr(field, ProdutoCorrente.embalagem);

    field = GetObjectPtrForm(frmP, IPFPrecoFld);
    FldSetTextPtr(field, preco);

    field = GetObjectPtrForm(frmP, IPFEstoqueFld);
    FldSetTextPtr(field, estoque);

}
```

soapclientes.c:

```
/* soapClient.c
   Generated by gSOAP 2.2.3b from wsdl.h
   Copyright (C) 2001-2003 Genivia inc.
   All Rights Reserved.
*/
#include "soapH.h"
#ifdef __cplusplus
extern "C" {
#endif

SOAP_SOURCE_STAMP("@(#) soapClient.c ver 2.2.3b 2004-02-03 03:25:16
GMT")


SOAP_FMAC1 int SOAP_FMAC2 soap_call_tns__addProdutoPedido(struct soap
*soap, const char *URL, const char *action, int pedidoID, int
produtoID, int quantidade, struct tns__addProdutoPedidoResponse *out)
{
      struct tns__addProdutoPedido soap_tmp_tns__addProdutoPedido;
      if (!action)
            action = "urn:Servico#servico#addProdutoPedido";
      soap_tmp_tns__addProdutoPedido.pedidoID=pedidoID;
      soap_tmp_tns__addProdutoPedido.produtoID=produtoID;
      soap_tmp_tns__addProdutoPedido.quantidade=quantidade;
      soap_begin(soap);
      soap_serializeheader(soap);
      soap_serialize_tns__addProdutoPedido(soap,
&soap_tmp_tns__addProdutoPedido);
      soap_begin_count(soap);
      if (soap->mode & SOAP_IO_LENGTH)
      {     soap_envelope_begin_out(soap);
            soap_putheader(soap);
            soap_body_begin_out(soap);
            soap_put_tns__addProdutoPedido(soap,
&soap_tmp_tns__addProdutoPedido, "tns:addProdutoPedido", "");
            soap_body_end_out(soap);
            soap_envelope_end_out(soap);
      }
      if (soap_connect(soap, URL, action)
       || soap_envelope_begin_out(soap)
       || soap_putheader(soap)
```

```
       || soap_body_begin_out(soap)
       || soap_put_tns__addProdutoPedido(soap,
&soap_tmp_tns__addProdutoPedido, "tns:addProdutoPedido", "")
       || soap_body_end_out(soap)
       || soap_envelope_end_out(soap)
       || soap_putattachments(soap)
       || soap_end_send(soap))
            return soap->error;
      soap_default_tns__addProdutoPedidoResponse(soap, out);
      if (soap_begin_recv(soap)
       || soap_envelope_begin_in(soap)
       || soap_recv_header(soap)
       || soap_body_begin_in(soap))
            return soap->error;
      soap_get_tns__addProdutoPedidoResponse(soap, out,
"tns:addProdutoPedidoResponse", "tns:addProdutoPedidoResponse");
      if (soap->error)
      {     if (soap->error == SOAP_TAG_MISMATCH && soap->level == 2)
                  soap_recv_fault(soap);
            return soap->error;
      }
      if (soap_body_end_in(soap)
       || soap_envelope_end_in(soap)
       || soap_getattachments(soap)
       || soap_end_recv(soap))
            return soap->error;
      soap_closesock(soap);
      return SOAP_OK;
}

SOAP_FMAC1 int SOAP_FMAC2 soap_call_tns__getInfoProduto(struct soap
*soap, const char *URL, const char *action, int produtoID, struct
tns__getInfoProdutoResponse *out)
{
      struct tns__getInfoProduto soap_tmp_tns__getInfoProduto;
      if (!action)
            action = "urn:Servico#servico#getInfoProduto";
      soap_tmp_tns__getInfoProduto.produtoID=produtoID;
      soap_begin(soap);
      soap_serializeheader(soap);
      soap_serialize_tns__getInfoProduto(soap,
&soap_tmp_tns__getInfoProduto);
      soap_begin_count(soap);
      if (soap->mode & SOAP_IO_LENGTH)
      {     soap_envelope_begin_out(soap);
            soap_putheader(soap);
            soap_body_begin_out(soap);
            soap_put_tns__getInfoProduto(soap,
&soap_tmp_tns__getInfoProduto, "tns:getInfoProduto", "");
            soap_body_end_out(soap);
            soap_envelope_end_out(soap);
      }
      if (soap_connect(soap, URL, action)
       || soap_envelope_begin_out(soap)
       || soap_putheader(soap)
       || soap_body_begin_out(soap)
       || soap_put_tns__getInfoProduto(soap,
&soap_tmp_tns__getInfoProduto, "tns:getInfoProduto", "")
       || soap_body_end_out(soap)
       || soap_envelope_end_out(soap)
       || soap_putattachments(soap)
       || soap_end_send(soap))
            return soap->error;
```

```
	soap_default_tns__getInfoProdutoResponse(soap, out);
	if (soap_begin_recv(soap)
	 || soap_envelope_begin_in(soap)
	 || soap_recv_header(soap)
	 || soap_body_begin_in(soap))
		return soap->error;
	soap_get_tns__getInfoProdutoResponse(soap, out,
"tns:getInfoProdutoResponse", "tns:getInfoProdutoResponse");
	if (soap->error)
	{	if (soap->error == SOAP_TAG_MISMATCH && soap->level == 2)
			soap_recv_fault(soap);
		return soap->error;
	}
	if (soap_body_end_in(soap)
	 || soap_envelope_end_in(soap)
	 || soap_getattachments(soap)
	 || soap_end_recv(soap))
		return soap->error;
	soap_closesock(soap);
	return SOAP_OK;
}

SOAP_FMAC1 int SOAP_FMAC2 soap_call_tns__getPedido(struct soap *soap,
const char *URL, const char *action, int clienteID, int vendedorID,
char *senha, struct tns__getPedidoResponse *out)
{
	struct tns__getPedido soap_tmp_tns__getPedido;
	if (!action)
		action = "urn:Servico#servico#getPedido";
	soap_tmp_tns__getPedido.clienteID=clienteID;
	soap_tmp_tns__getPedido.vendedorID=vendedorID;
	soap_tmp_tns__getPedido.senha=senha;
	soap_begin(soap);
	soap_serializeheader(soap);
	soap_serialize_tns__getPedido(soap, &soap_tmp_tns__getPedido);
	soap_begin_count(soap);
	if (soap->mode & SOAP_IO_LENGTH)
	{	soap_envelope_begin_out(soap);
		soap_putheader(soap);
		soap_body_begin_out(soap);
		soap_put_tns__getPedido(soap, &soap_tmp_tns__getPedido,
"tns:getPedido", "");
		soap_body_end_out(soap);
		soap_envelope_end_out(soap);
	}
	if (soap_connect(soap, URL, action)
	 || soap_envelope_begin_out(soap)
	 || soap_putheader(soap)
	 || soap_body_begin_out(soap)
	 || soap_put_tns__getPedido(soap, &soap_tmp_tns__getPedido,
"tns:getPedido", "")
	 || soap_body_end_out(soap)
	 || soap_envelope_end_out(soap)
	 || soap_putattachments(soap)
	 || soap_end_send(soap))
		return soap->error;
	soap_default_tns__getPedidoResponse(soap, out);
	if (soap_begin_recv(soap)
	 || soap_envelope_begin_in(soap)
	 || soap_recv_header(soap)
	 || soap_body_begin_in(soap))
		return soap->error;
```

```
      soap_get_tns__getPedidoResponse(soap, out,
"tns:getPedidoResponse", "tns:getPedidoResponse");
      if (soap->error)
      {     if (soap->error == SOAP_TAG_MISMATCH && soap->level == 2)
                  soap_recv_fault(soap);
            return soap->error;
      }
      if (soap_body_end_in(soap)
       || soap_envelope_end_in(soap)
       || soap_getattachments(soap)
       || soap_end_recv(soap))
            return soap->error;
      soap_closesock(soap);
      return SOAP_OK;
}

SOAP_FMAC1 int SOAP_FMAC2 soap_call_tns__getEstoque(struct soap *soap,
const char *URL, const char *action, int produtoID, struct
tns__getEstoqueResponse *out)
{
      struct tns__getEstoque soap_tmp_tns__getEstoque;
      if (!action)
            action = "urn:Servico#servico#getEstoque";
      soap_tmp_tns__getEstoque.produtoID=produtoID;
      soap_begin(soap);
      soap_serializeheader(soap);
      soap_serialize_tns__getEstoque(soap, &soap_tmp_tns__getEstoque);
      soap_begin_count(soap);
      if (soap->mode & SOAP_IO_LENGTH)
      {     soap_envelope_begin_out(soap);
            soap_putheader(soap);
            soap_body_begin_out(soap);
            soap_put_tns__getEstoque(soap, &soap_tmp_tns__getEstoque,
"tns:getEstoque", "");
            soap_body_end_out(soap);
            soap_envelope_end_out(soap);
      }
      if (soap_connect(soap, URL, action)
       || soap_envelope_begin_out(soap)
       || soap_putheader(soap)
       || soap_body_begin_out(soap)
       || soap_put_tns__getEstoque(soap, &soap_tmp_tns__getEstoque,
"tns:getEstoque", "")
       || soap_body_end_out(soap)
       || soap_envelope_end_out(soap)
       || soap_putattachments(soap)
       || soap_end_send(soap))
            return soap->error;
      soap_default_tns__getEstoqueResponse(soap, out);
      if (soap_begin_recv(soap)
       || soap_envelope_begin_in(soap)
       || soap_recv_header(soap)
       || soap_body_begin_in(soap))
            return soap->error;
      soap_get_tns__getEstoqueResponse(soap, out,
"tns:getEstoqueResponse", "tns:getEstoqueResponse");
      if (soap->error)
      {     if (soap->error == SOAP_TAG_MISMATCH && soap->level == 2)
                  soap_recv_fault(soap);
            return soap->error;
      }
      if (soap_body_end_in(soap)
       || soap_envelope_end_in(soap)
```

```
       || soap_getattachments(soap)
       || soap_end_recv(soap))
            return soap->error;
      soap_closesock(soap);
      return SOAP_OK;
}

SOAP_FMAC1 int SOAP_FMAC2 soap_call_tns__getInfoCliente(struct soap
*soap, const char *URL, const char *action, int clienteID, struct
tns__getInfoClienteResponse *out)
{
      struct tns__getInfoCliente soap_tmp_tns__getInfoCliente;
      if (!action)
            action = "urn:Servico#servico#getInfoCliente";
      soap_tmp_tns__getInfoCliente.clienteID=clienteID;
      soap_begin(soap);
      soap_serializeheader(soap);
      soap_serialize_tns__getInfoCliente(soap,
&soap_tmp_tns__getInfoCliente);
      soap_begin_count(soap);
      if (soap->mode & SOAP_IO_LENGTH)
      {     soap_envelope_begin_out(soap);
            soap_putheader(soap);
            soap_body_begin_out(soap);
            soap_put_tns__getInfoCliente(soap,
&soap_tmp_tns__getInfoCliente, "tns:getInfoCliente", "");
            soap_body_end_out(soap);
            soap_envelope_end_out(soap);
      }
      if (soap_connect(soap, URL, action)
       || soap_envelope_begin_out(soap)
       || soap_putheader(soap)
       || soap_body_begin_out(soap)
       || soap_put_tns__getInfoCliente(soap,
&soap_tmp_tns__getInfoCliente, "tns:getInfoCliente", "")
       || soap_body_end_out(soap)
       || soap_envelope_end_out(soap)
       || soap_putattachments(soap)
       || soap_end_send(soap))
            return soap->error;
      soap_default_tns__getInfoClienteResponse(soap, out);
      if (soap_begin_recv(soap)
       || soap_envelope_begin_in(soap)
       || soap_recv_header(soap)
       || soap_body_begin_in(soap))
            return soap->error;
      soap_get_tns__getInfoClienteResponse(soap, out,
"tns:getInfoClienteResponse", "tns:getInfoClienteResponse");
      if (soap->error)
      {     if (soap->error == SOAP_TAG_MISMATCH && soap->level == 2)
                  soap_recv_fault(soap);
            return soap->error;
      }
      if (soap_body_end_in(soap)
       || soap_envelope_end_in(soap)
       || soap_getattachments(soap)
       || soap_end_recv(soap))
            return soap->error;
      soap_closesock(soap);
      return SOAP_OK;
}
```

```
SOAP_FMAC1 int SOAP_FMAC2 soap_call_tns__getComissao(struct soap
*soap, const char *URL, const char *action, int vendedorID, char
*senha, struct tns__getComissaoResponse *out)
{
      struct tns__getComissao soap_tmp_tns__getComissao;
      if (!action)
            action = "urn:Servico#servico#getComissao";
      soap_tmp_tns__getComissao.vendedorID=vendedorID;
      soap_tmp_tns__getComissao.senha=senha;
      soap_begin(soap);
      soap_serializeheader(soap);
      soap_serialize_tns__getComissao(soap,
&soap_tmp_tns__getComissao);
      soap_begin_count(soap);
      if (soap->mode & SOAP_IO_LENGTH)
      {     soap_envelope_begin_out(soap);
            soap_putheader(soap);
            soap_body_begin_out(soap);
            soap_put_tns__getComissao(soap,
&soap_tmp_tns__getComissao, "tns:getComissao", "");
            soap_body_end_out(soap);
            soap_envelope_end_out(soap);
      }
      if (soap_connect(soap, URL, action)
       || soap_envelope_begin_out(soap)
       || soap_putheader(soap)
       || soap_body_begin_out(soap)
       || soap_put_tns__getComissao(soap, &soap_tmp_tns__getComissao,
"tns:getComissao", "")
       || soap_body_end_out(soap)
       || soap_envelope_end_out(soap)
       || soap_putattachments(soap)
       || soap_end_send(soap))
            return soap->error;
      soap_default_tns__getComissaoResponse(soap, out);
      if (soap_begin_recv(soap)
       || soap_envelope_begin_in(soap)
       || soap_recv_header(soap)
       || soap_body_begin_in(soap))
            return soap->error;
      soap_get_tns__getComissaoResponse(soap, out,
"tns:getComissaoResponse", "tns:getComissaoResponse");
      if (soap->error)
      {     if (soap->error == SOAP_TAG_MISMATCH && soap->level == 2)
                  soap_recv_fault(soap);
            return soap->error;
      }
      if (soap_body_end_in(soap)
       || soap_envelope_end_in(soap)
       || soap_getattachments(soap)
       || soap_end_recv(soap))
            return soap->error;
      soap_closesock(soap);
      return SOAP_OK;
}
#ifdef __cplusplus
}
#endif

/* end of soapClient.c */
```

soapserv.c:

```
/* soapServer.c
   Generated by gSOAP 2.2.3b from wsdl.h
   Copyright (C) 2001-2003 Genivia inc.
   All Rights Reserved.
*/
#include "soapH.h"
#ifdef __cplusplus
extern "C" {
#endif

SOAP_SOURCE_STAMP("@(#) soapServer.c ver 2.2.3b 2004-02-03 03:25:16
GMT")


SOAP_FMAC1 int SOAP_FMAC2 soap_serve(struct soap *soap)
{
      unsigned int n = SOAP_MAXKEEPALIVE;
      do
      {     soap_begin(soap);
            if (soap_begin_recv(soap) || soap_envelope_begin_in(soap)
|| soap_recv_header(soap) || soap_body_begin_in(soap))
                  return soap_send_fault(soap);
            if (!--n)
                  soap->keep_alive = 0;
            soap->error = soap_serve_tns__addProdutoPedido(soap);
            if (soap->error == SOAP_NO_METHOD)
                  soap_serve_tns__getInfoProduto(soap);
            if (soap->error == SOAP_NO_METHOD)
                  soap_serve_tns__getPedido(soap);
            if (soap->error == SOAP_NO_METHOD)
                  soap_serve_tns__getEstoque(soap);
            if (soap->error == SOAP_NO_METHOD)
                  soap_serve_tns__getInfoCliente(soap);
            if (soap->error == SOAP_NO_METHOD)
                  soap_serve_tns__getComissao(soap);
            if (soap->error)
                  return soap_send_fault(soap);
      } while (soap->keep_alive);
      return SOAP_OK;
}

SOAP_FMAC1 int SOAP_FMAC2 soap_serve_tns__addProdutoPedido(struct soap
*soap)
{     struct tns__addProdutoPedido soap_tmp_tns__addProdutoPedido;
      struct tns__addProdutoPedidoResponse out;
      soap_default_tns__addProdutoPedidoResponse(soap, &out);
      soap_default_tns__addProdutoPedido(soap,
&soap_tmp_tns__addProdutoPedido);
      soap_get_tns__addProdutoPedido(soap,
&soap_tmp_tns__addProdutoPedido, "tns:addProdutoPedido", NULL);
      if (soap->error == SOAP_TAG_MISMATCH && soap->level == 2)
            soap->error = SOAP_NO_METHOD;
      if (soap->error)
            return soap->error;

      if (soap_body_end_in(soap)
       || soap_envelope_end_in(soap)
       || soap_getattachments(soap)
       || soap_end_recv(soap))
            return soap->error;
```

```
      soap->error = tns__addProdutoPedido(soap,
soap_tmp_tns__addProdutoPedido.pedidoID,
soap_tmp_tns__addProdutoPedido.produtoID,
soap_tmp_tns__addProdutoPedido.quantidade, &out);
      if (soap->error)
            return soap->error;
      soap_serializeheader(soap);
      soap_serialize_tns__addProdutoPedidoResponse(soap, &out);
      soap_begin_count(soap);
      if (soap->mode & SOAP_IO_LENGTH)
      {     soap_envelope_begin_out(soap);
            soap_putheader(soap);
            soap_body_begin_out(soap);
            soap_put_tns__addProdutoPedidoResponse(soap, &out,
"tns:addProdutoPedidoResponse", "");
            soap_body_end_out(soap);
            soap_envelope_end_out(soap);
      };
      if (soap_response(soap, SOAP_OK)
       || soap_envelope_begin_out(soap)
       || soap_putheader(soap)
       || soap_body_begin_out(soap)
       || soap_put_tns__addProdutoPedidoResponse(soap, &out,
"tns:addProdutoPedidoResponse", "")
       || soap_body_end_out(soap)
       || soap_envelope_end_out(soap)
       || soap_putattachments(soap)
       || soap_end_send(soap))
            return soap->error;
      soap_closesock(soap);
      return SOAP_OK;
}

SOAP_FMAC1 int SOAP_FMAC2 soap_serve_tns__getInfoProduto(struct soap
*soap)
{     struct tns__getInfoProduto soap_tmp_tns__getInfoProduto;
      struct tns__getInfoProdutoResponse out;
      soap_default_tns__getInfoProdutoResponse(soap, &out);
      soap_default_tns__getInfoProduto(soap,
&soap_tmp_tns__getInfoProduto);
      soap_get_tns__getInfoProduto(soap,
&soap_tmp_tns__getInfoProduto, "tns:getInfoProduto", NULL);
      if (soap->error == SOAP_TAG_MISMATCH && soap->level == 2)
            soap->error = SOAP_NO_METHOD;
      if (soap->error)
            return soap->error;

      if (soap_body_end_in(soap)
       || soap_envelope_end_in(soap)
       || soap_getattachments(soap)
       || soap_end_recv(soap))
            return soap->error;
      soap->error = tns__getInfoProduto(soap,
soap_tmp_tns__getInfoProduto.produtoID, &out);
      if (soap->error)
            return soap->error;
      soap_serializeheader(soap);
      soap_serialize_tns__getInfoProdutoResponse(soap, &out);
      soap_begin_count(soap);
      if (soap->mode & SOAP_IO_LENGTH)
      {     soap_envelope_begin_out(soap);
            soap_putheader(soap);
            soap_body_begin_out(soap);
```

```
            soap_put_tns__getInfoProdutoResponse(soap, &out,
"tns:getInfoProdutoResponse", "");
            soap_body_end_out(soap);
            soap_envelope_end_out(soap);
      };
      if (soap_response(soap, SOAP_OK)
       || soap_envelope_begin_out(soap)
       || soap_putheader(soap)
       || soap_body_begin_out(soap)
       || soap_put_tns__getInfoProdutoResponse(soap, &out,
"tns:getInfoProdutoResponse", "")
       || soap_body_end_out(soap)
       || soap_envelope_end_out(soap)
       || soap_putattachments(soap)
       || soap_end_send(soap))
            return soap->error;
      soap_closesock(soap);
      return SOAP_OK;
}

SOAP_FMAC1 int SOAP_FMAC2 soap_serve_tns__getPedido(struct soap *soap)
{     struct tns__getPedido soap_tmp_tns__getPedido;
      struct tns__getPedidoResponse out;
      soap_default_tns__getPedidoResponse(soap, &out);
      soap_default_tns__getPedido(soap, &soap_tmp_tns__getPedido);
      soap_get_tns__getPedido(soap, &soap_tmp_tns__getPedido,
"tns:getPedido", NULL);
      if (soap->error == SOAP_TAG_MISMATCH && soap->level == 2)
            soap->error = SOAP_NO_METHOD;
      if (soap->error)
            return soap->error;

      if (soap_body_end_in(soap)
       || soap_envelope_end_in(soap)
       || soap_getattachments(soap)
       || soap_end_recv(soap))
            return soap->error;
      soap->error = tns__getPedido(soap,
soap_tmp_tns__getPedido.clienteID, soap_tmp_tns__getPedido.vendedorID,
soap_tmp_tns__getPedido.senha, &out);
      if (soap->error)
            return soap->error;
      soap_serializeheader(soap);
      soap_serialize_tns__getPedidoResponse(soap, &out);
      soap_begin_count(soap);
      if (soap->mode & SOAP_IO_LENGTH)
      {     soap_envelope_begin_out(soap);
            soap_putheader(soap);
            soap_body_begin_out(soap);
            soap_put_tns__getPedidoResponse(soap, &out,
"tns:getPedidoResponse", "");
            soap_body_end_out(soap);
            soap_envelope_end_out(soap);
      };
      if (soap_response(soap, SOAP_OK)
       || soap_envelope_begin_out(soap)
       || soap_putheader(soap)
       || soap_body_begin_out(soap)
       || soap_put_tns__getPedidoResponse(soap, &out,
"tns:getPedidoResponse", "")
       || soap_body_end_out(soap)
       || soap_envelope_end_out(soap)
       || soap_putattachments(soap)
```

```
       || soap_end_send(soap))
            return soap->error;
      soap_closesock(soap);
      return SOAP_OK;
}

SOAP_FMAC1 int SOAP_FMAC2 soap_serve_tns__getEstoque(struct soap
*soap)
{     struct tns__getEstoque soap_tmp_tns__getEstoque;
      struct tns__getEstoqueResponse out;
      soap_default_tns__getEstoqueResponse(soap, &out);
      soap_default_tns__getEstoque(soap, &soap_tmp_tns__getEstoque);
      soap_get_tns__getEstoque(soap, &soap_tmp_tns__getEstoque,
"tns:getEstoque", NULL);
      if (soap->error == SOAP_TAG_MISMATCH && soap->level == 2)
            soap->error = SOAP_NO_METHOD;
      if (soap->error)
            return soap->error;

      if (soap_body_end_in(soap)
       || soap_envelope_end_in(soap)
       || soap_getattachments(soap)
       || soap_end_recv(soap))
            return soap->error;
      soap->error = tns__getEstoque(soap,
soap_tmp_tns__getEstoque.produtoID, &out);
      if (soap->error)
            return soap->error;
      soap_serializeheader(soap);
      soap_serialize_tns__getEstoqueResponse(soap, &out);
      soap_begin_count(soap);
      if (soap->mode & SOAP_IO_LENGTH)
      {     soap_envelope_begin_out(soap);
            soap_putheader(soap);
            soap_body_begin_out(soap);
            soap_put_tns__getEstoqueResponse(soap, &out,
"tns:getEstoqueResponse", "");
            soap_body_end_out(soap);
            soap_envelope_end_out(soap);
      };
      if (soap_response(soap, SOAP_OK)
       || soap_envelope_begin_out(soap)
       || soap_putheader(soap)
       || soap_body_begin_out(soap)
       || soap_put_tns__getEstoqueResponse(soap, &out,
"tns:getEstoqueResponse", "")
       || soap_body_end_out(soap)
       || soap_envelope_end_out(soap)
       || soap_putattachments(soap)
       || soap_end_send(soap))
            return soap->error;
      soap_closesock(soap);
      return SOAP_OK;
}

SOAP_FMAC1 int SOAP_FMAC2 soap_serve_tns__getInfoCliente(struct soap
*soap)
{     struct tns__getInfoCliente soap_tmp_tns__getInfoCliente;
      struct tns__getInfoClienteResponse out;
      soap_default_tns__getInfoClienteResponse(soap, &out);
      soap_default_tns__getInfoCliente(soap,
&soap_tmp_tns__getInfoCliente);
```

```
      soap_get_tns__getInfoCliente(soap,
&soap_tmp_tns__getInfoCliente, "tns:getInfoCliente", NULL);
      if (soap->error == SOAP_TAG_MISMATCH && soap->level == 2)
            soap->error = SOAP_NO_METHOD;
      if (soap->error)
            return soap->error;

      if (soap_body_end_in(soap)
       || soap_envelope_end_in(soap)
       || soap_getattachments(soap)
       || soap_end_recv(soap))
            return soap->error;
      soap->error = tns__getInfoCliente(soap,
soap_tmp_tns__getInfoCliente.clienteID, &out);
      if (soap->error)
            return soap->error;
      soap_serializeheader(soap);
      soap_serialize_tns__getInfoClienteResponse(soap, &out);
      soap_begin_count(soap);
      if (soap->mode & SOAP_IO_LENGTH)
      {     soap_envelope_begin_out(soap);
            soap_putheader(soap);
            soap_body_begin_out(soap);
            soap_put_tns__getInfoClienteResponse(soap, &out,
"tns:getInfoClienteResponse", "");
            soap_body_end_out(soap);
            soap_envelope_end_out(soap);
      };
      if (soap_response(soap, SOAP_OK)
       || soap_envelope_begin_out(soap)
       || soap_putheader(soap)
       || soap_body_begin_out(soap)
       || soap_put_tns__getInfoClienteResponse(soap, &out,
"tns:getInfoClienteResponse", "")
       || soap_body_end_out(soap)
       || soap_envelope_end_out(soap)
       || soap_putattachments(soap)
       || soap_end_send(soap))
            return soap->error;
      soap_closesock(soap);
      return SOAP_OK;
}

SOAP_FMAC1 int SOAP_FMAC2 soap_serve_tns__getComissao(struct soap
*soap)
{     struct tns__getComissao soap_tmp_tns__getComissao;
      struct tns__getComissaoResponse out;
      soap_default_tns__getComissaoResponse(soap, &out);
      soap_default_tns__getComissao(soap, &soap_tmp_tns__getComissao);
      soap_get_tns__getComissao(soap, &soap_tmp_tns__getComissao,
"tns:getComissao", NULL);
      if (soap->error == SOAP_TAG_MISMATCH && soap->level == 2)
            soap->error = SOAP_NO_METHOD;
      if (soap->error)
            return soap->error;

      if (soap_body_end_in(soap)
       || soap_envelope_end_in(soap)
       || soap_getattachments(soap)
       || soap_end_recv(soap))
            return soap->error;
```

```
      soap->error = tns__getComissao(soap,
soap_tmp_tns__getComissao.vendedorID, soap_tmp_tns__getComissao.senha,
&out);
      if (soap->error)
            return soap->error;
      soap_serializeheader(soap);
      soap_serialize_tns__getComissaoResponse(soap, &out);
      soap_begin_count(soap);
      if (soap->mode & SOAP_IO_LENGTH)
      {     soap_envelope_begin_out(soap);
            soap_putheader(soap);
            soap_body_begin_out(soap);
            soap_put_tns__getComissaoResponse(soap, &out,
"tns:getComissaoResponse", "");
            soap_body_end_out(soap);
            soap_envelope_end_out(soap);
      };
      if (soap_response(soap, SOAP_OK)
       || soap_envelope_begin_out(soap)
       || soap_putheader(soap)
       || soap_body_begin_out(soap)
       || soap_put_tns__getComissaoResponse(soap, &out,
"tns:getComissaoResponse", "")
       || soap_body_end_out(soap)
       || soap_envelope_end_out(soap)
       || soap_putattachments(soap)
       || soap_end_send(soap))
            return soap->error;
      soap_closesock(soap);
      return SOAP_OK;
}
#ifdef __cplusplus
}
#endif

/* end of soapServer.c */
```

soapsoap.h:

```
/* soapsoapServicoServiceProxy.h
   Generated by gSOAP 2.2.3b from wsdl.h
   Copyright (C) 2001-2003 Genivia inc.
   All Rights Reserved.
*/

#ifndef soapServicoService_H
#define soapServicoService_H
#include "soapH.h"
class soapServicoService
{   public:
      struct soap *soap;
      const char *endpoint;
      soapServicoService() { soap = soap_new(); endpoint =
"http://grumiche.ath.cx:8080/service/servidor.php"; };
      ~soapServicoService() { if (soap) { soap_destroy(soap); soap_end
(soap); soap_done(soap); free((void*)soap); } };
      int addProdutoPedido(int pedidoID, int produtoID, int
quantidade, struct tns__addProdutoPedidoResponse *out) { return soap ?
soap_call_tns__addProdutoPedido(soap, endpoint, NULL, pedidoID,
produtoID, quantidade, out) : SOAP_EOM; };
      int getInfoProduto(int produtoID, struct
tns__getInfoProdutoResponse *out) { return soap ?
```

```
soap_call_tns__getInfoProduto(soap, endpoint, NULL, produtoID, out) :
SOAP_EOM; };
      int getPedido(int clienteID, int vendedorID, char *senha, struct
tns__getPedidoResponse *out) { return soap ? soap_call_tns__getPedido
(soap, endpoint, NULL, clienteID, vendedorID, senha, out) :
SOAP_EOM; };
      int getEstoque(int produtoID, struct tns__getEstoqueResponse
*out) { return soap ? soap_call_tns__getEstoque(soap, endpoint, NULL,
produtoID, out) : SOAP_EOM; };
      int getInfoCliente(int clienteID, struct
tns__getInfoClienteResponse *out) { return soap ?
soap_call_tns__getInfoCliente(soap, endpoint, NULL, clienteID, out) :
SOAP_EOM; };
      int getComissao(int vendedorID, char *senha, struct
tns__getComissaoResponse *out) { return soap ?
soap_call_tns__getComissao(soap, endpoint, NULL, vendedorID, senha,
out) : SOAP_EOM; };
};
#endif
```

soapstub.h:

```
/* soapStub.h
   Generated by gSOAP 2.2.3b from wsdl.h
   Copyright (C) 2001-2003 Genivia inc.
   All Rights Reserved.
*/
#ifndef soapStub_H
#define soapStub_H
#ifdef __cplusplus
extern "C" {
#endif

#define SOAP_SECTION  __attribute__ ((section ("soap")))

//#define SOAP_SECTION_20  __attribute__ ((section ("soap20")))
//#define SOAP_SECTION_21  __attribute__ ((section ("soap21")))
//#define SOAP_SECTION_22  __attribute__ ((section ("soap22")))
//#define SOAP_SECTION_23  __attribute__ ((section ("soap23")))
//#define SOAP_SECTION_24  __attribute__ ((section ("soap24")))
//#define SOAP_SECTION_25  __attribute__ ((section ("soap25")))
//#define SOAP_SECTION_26  __attribute__ ((section ("soap26")))
//#define SOAP_SECTION_27  __attribute__ ((section ("soap27")))
//#define SOAP_SECTION_28  __attribute__ ((section ("soap28")))
//#define SOAP_SECTION_29  __attribute__ ((section ("soap29")))
//#define SOAP_SECTION_30  __attribute__ ((section ("soap30")))
//#define SOAP_SECTION_31  __attribute__ ((section ("soap31")))



/* Types With Custom (De)serializers: */

/* Enumerations */

#ifndef _SOAP_Enum_
#define _SOAP_Enum_
enum Enum_ {false_ = 0, true_ = 1};
#endif

/* Classes and Structs */

#ifndef _SOAP_tns__InfoCliente
```

```
#define _SOAP_tns__InfoCliente
struct tns__InfoCliente
{
      int clienteID;
      char *nome;
      char *endereco;
      char *bairro;
      char *cidade;
      char *estado;
};
#endif

#ifndef _SOAP_tns__getInfoClienteResponse
#define _SOAP_tns__getInfoClienteResponse
struct tns__getInfoClienteResponse
{
      struct tns__InfoCliente *_infoCliente;
};
#endif

#ifndef _SOAP_tns__InfoProduto
#define _SOAP_tns__InfoProduto
struct tns__InfoProduto
{
      int produtoID;
      char *descricao;
      float preco;
      int estoque;
};
#endif

#ifndef _SOAP_tns__getInfoProdutoResponse
#define _SOAP_tns__getInfoProdutoResponse
struct tns__getInfoProdutoResponse
{
      struct tns__InfoProduto *_infoProduto;
};
#endif

#ifndef _SOAP_tns__addProdutoPedidoResponse
#define _SOAP_tns__addProdutoPedidoResponse
struct tns__addProdutoPedidoResponse
{
      enum Enum_ _adicionado;
};
#endif

#ifndef _SOAP_tns__getEstoqueResponse
#define _SOAP_tns__getEstoqueResponse
struct tns__getEstoqueResponse
{
      int _quantidade;
};
#endif

#ifndef _SOAP_tns__getComissaoResponse
#define _SOAP_tns__getComissaoResponse
struct tns__getComissaoResponse
{
      float _comissao;
};
#endif
```

```
#ifndef _SOAP_tns__getPedidoResponse
#define _SOAP_tns__getPedidoResponse
struct tns__getPedidoResponse
{
      int _pedidoID;
};
#endif

#ifndef _SOAP_tns__addProdutoPedido
#define _SOAP_tns__addProdutoPedido
struct tns__addProdutoPedido
{
      int pedidoID;
      int produtoID;
      int quantidade;
};
#endif

#ifndef _SOAP_tns__getInfoProduto
#define _SOAP_tns__getInfoProduto
struct tns__getInfoProduto
{
      int produtoID;
};
#endif

#ifndef _SOAP_tns__getPedido
#define _SOAP_tns__getPedido
struct tns__getPedido
{
      int clienteID;
      int vendedorID;
      char *senha;
};
#endif

#ifndef _SOAP_tns__getEstoque
#define _SOAP_tns__getEstoque
struct tns__getEstoque
{
      int produtoID;
};
#endif

#ifndef _SOAP_tns__getInfoCliente
#define _SOAP_tns__getInfoCliente
struct tns__getInfoCliente
{
      int clienteID;
};
#endif

#ifndef _SOAP_tns__getComissao
#define _SOAP_tns__getComissao
struct tns__getComissao
{
      int vendedorID;
      char *senha;
};
#endif

#ifndef _SOAP_SOAP_ENV__Header
#define _SOAP_SOAP_ENV__Header
```

```
/* SOAP Header: */
struct SOAP_ENV__Header
{
      void *dummy;      /* transient */
};
#endif

#ifndef _SOAP_SOAP_ENV__Code
#define _SOAP_SOAP_ENV__Code
/* SOAP Fault Code: */
struct SOAP_ENV__Code
{
      char *SOAP_ENV__Value;
      char *SOAP_ENV__Node;
      char *SOAP_ENV__Role;
};
#endif

#ifndef _SOAP_SOAP_ENV__Fault
#define _SOAP_SOAP_ENV__Fault
/* SOAP Fault: */
struct SOAP_ENV__Fault
{
      char *faultcode;
      char *faultstring;
      char *faultactor;
      char *detail;
      struct SOAP_ENV__Code *SOAP_ENV__Code;
      char *SOAP_ENV__Reason;
      char *SOAP_ENV__Detail;
};
#endif

/* Typedefs */
typedef char *xsd__string;
typedef int xsd__int;
typedef enum Enum_ xsd__boolean;
typedef float xsd__float;

/* Variables */

/* Remote Methods */

SOAP_FMAC1 int SOAP_FMAC2 tns__addProdutoPedido(struct soap*, int,
int, int, struct tns__addProdutoPedidoResponse *)SOAP_SECTION;

SOAP_FMAC1 int SOAP_FMAC2 tns__getInfoProduto(struct soap*, int,
struct tns__getInfoProdutoResponse *)SOAP_SECTION;

SOAP_FMAC1 int SOAP_FMAC2 tns__getPedido(struct soap*, int, int, char
*, struct tns__getPedidoResponse *)SOAP_SECTION;

SOAP_FMAC1 int SOAP_FMAC2 tns__getEstoque(struct soap*, int, struct
tns__getEstoqueResponse *)SOAP_SECTION;

SOAP_FMAC1 int SOAP_FMAC2 tns__getInfoCliente(struct soap*, int,
struct tns__getInfoClienteResponse *)SOAP_SECTION;

SOAP_FMAC1 int SOAP_FMAC2 tns__getComissao(struct soap*, int, char *,
struct tns__getComissaoResponse *)SOAP_SECTION;

/* Stubs */
```

```
SOAP_FMAC1 int SOAP_FMAC2 soap_call_tns__addProdutoPedido(struct
soap*, const char*, const char*, int, int, int, struct
tns__addProdutoPedidoResponse *)SOAP_SECTION;

SOAP_FMAC1 int SOAP_FMAC2 soap_call_tns__getInfoProduto(struct soap*,
const char*, const char*, int, struct tns__getInfoProdutoResponse *)
SOAP_SECTION;

SOAP_FMAC1 int SOAP_FMAC2 soap_call_tns__getPedido(struct soap*, const
char*, const char*, int, int, char *, struct tns__getPedidoResponse *)
SOAP_SECTION;

SOAP_FMAC1 int SOAP_FMAC2 soap_call_tns__getEstoque(struct soap*,
const char*, const char*, int, struct tns__getEstoqueResponse *)
SOAP_SECTION;

SOAP_FMAC1 int SOAP_FMAC2 soap_call_tns__getInfoCliente(struct soap*,
const char*, const char*, int, struct tns__getInfoClienteResponse *)
SOAP_SECTION;

SOAP_FMAC1 int SOAP_FMAC2 soap_call_tns__getComissao(struct soap*,
const char*, const char*, int, char *, struct tns__getComissaoResponse
*)SOAP_SECTION;

/* Skeletons */

SOAP_FMAC1 int SOAP_FMAC2 soap_serve(struct soap*)SOAP_SECTION;

SOAP_FMAC1 int SOAP_FMAC2 soap_serve_tns__addProdutoPedido(struct
soap*)SOAP_SECTION;

SOAP_FMAC1 int SOAP_FMAC2 soap_serve_tns__getInfoProduto(struct soap*)
SOAP_SECTION;

SOAP_FMAC1 int SOAP_FMAC2 soap_serve_tns__getPedido(struct soap*)
SOAP_SECTION;

SOAP_FMAC1 int SOAP_FMAC2 soap_serve_tns__getEstoque(struct soap*)
SOAP_SECTION;

SOAP_FMAC1 int SOAP_FMAC2 soap_serve_tns__getInfoCliente(struct soap*)
SOAP_SECTION;

SOAP_FMAC1 int SOAP_FMAC2 soap_serve_tns__getComissao(struct soap*)
SOAP_SECTION;
#ifdef __cplusplus
}
#endif
#endif

/* end of soapStub.h */
```

util.c:

```
#include <PalmOS.h>

void* GetObjectPtr(UInt16 objectID);

void* GetObjectPtrForm(FormPtr frmP, UInt16 objectID);


void* GetObjectPtr(UInt16 objectID)
```

```
{
      FormPtr frmP;

      frmP = FrmGetActiveForm();
      return FrmGetObjectPtr(frmP, FrmGetObjectIndex(frmP, objectID));
}

void* GetObjectPtrForm(FormPtr frmP, UInt16 objectID) {
      return FrmGetObjectPtr(frmP, FrmGetObjectIndex(frmP, objectID));
}
```

vendas.c:

```
#include "Vendas.h"
#include "vendasprc.h"
#include "VendasTypes.h"
#include "VendasDbs.h"
//#include "soapH.h"
//#include "soapStub.h"
//#include "palmSharedLib.h"
//#include "stdsoap2.h"
#include "ErrorMgr.h"



/* Variáveis globais do sistema ... */

ClienteType ClienteCorrente;
RamoType    RamoCorrente;

//struct soap *SoapEngine;



UInt32 PilotMain(UInt16 cmd, MemPtr cmdPBP, UInt16 launchFlags)
{
    return StarterPalmMain(cmd, cmdPBP, launchFlags);
}



/** Função principal do programa.
 * Inicia a execução do programa, verificando o launch code informado
 * pelo Palm OS.#include "ErrorMgr.h"

 *
 * @param cmd Launch Code
 * @param cmdPBP Buffer contendo estrutura com dados relacionados com
o Lauch Code
 * @param Flags relativas ao lauch code.
 */
static UInt32 StarterPalmMain(UInt16 cmd, MemPtr cmdPBP, UInt16
launchFlags) {

    Err error;
    /* verifica se a versão do Palm OS é suportada pelo nosso programa
*/
      error = RomVersionCompatible (ourMinVersion, launchFlags);
      if (error) return error;

      /* agora verificamos qual o launch code enviado */
```

```
      switch (cmd)
    {
        case sysAppLaunchCmdNormalLaunch:
             if ((error = StartApp())) return error;       /* inicia
a aplicação */
            FrmGotoForm(SelClienteForm); /* abre o form principal */
            EventLoop();            /* faz aplicação entrar em loop a
espera de eventos */
              StopApp();     /* pára a aplicação */
            break;
        default:
            break;
    }
    return errNone; /*constante que indica que não houve erro*/
}



/** Inicia a aplicação.
 * Inicia a aplicação, realizando todas as ações necessárias para que
 * o programa possa enfim executa #EventLoop. Em caso de falha, é
retornado um
 * código de erro diferente de zero.
 *
 *
 * @return Zero em caso de sucesso, em caso contrário um código de
erro.
 */

static Err StartApp(void)
{
    Err erro;
      StarterPreferenceType prefs;
      UInt16 prefsSize;
      /* Lê as configurações salvas da aplicação e também informações
de estado */


    prefsSize = sizeof(StarterPreferenceType);
      if (PrefGetAppPreferences(appFileCreator, appPrefID, &prefs,
&prefsSize, true) !=
            noPreferenceFound)
      {
        /* Inserir instruções de inicalização da aplicação */
      }

    //erro = RamoDbAbre();
    //ErrFatalDisplayIf(erro, "Erro ao abrir RamoDb");
    ClienteCorrente.clienteId = 0;
    RamoCorrente.ramoId = 0;

    erro = ClienteDbAbre();
    if (erro != errNone) {
        SysFatalAlert(" Não foi possível abrir o bando de dados de
Cliente");
    }

    erro = ProdutoDbAbre();
    if (erro != errNone) {
        SysFatalAlert(" Não foi possível abrir o bando de dados de
Cliente");
    }
```

```
    //SoapEngine = soap_new();

      return errNone;
}



/** Pára a aplicação
 *
 */
static void StopApp(void)
{
      StarterPreferenceType prefs;

//    SysFatalAlert("StopApp chamado");

      /* Grava as preferências/configurações da aplicação, assim como
infomações
        de estado e demais quaisquer que o programador desejar :-)
       */
      PrefSetAppPreferences (appFileCreator, appPrefID,
appPrefVersionNum,
            &prefs, sizeof (prefs), true);

    /* Fecha todos os forms */
      FrmCloseAllForms();
    //    RamoDbFecha();
    ClienteDbFecha();
    ProdutoDbFecha();
    //soap_end(SoapEngine);
    //free(SoapEngine);

}




/** Inicia o loop de captação de eventos para a aplicação.
 *
 * Implementa um loop que capta todos os eventos enviados para a mesma
e
 * delega o processamento destes. Se o evento appStopEvent é recebido,
é encerrado
 * o loop e a função termina sua execução. <p>
 * O evento appStopEvent indica que o programa deverá ser terminado.
 *
 */
static void EventLoop(void)
{
    EventType event;
    Err error;

    do
    {
        EvtGetEvent(&event, evtWaitForever);
        if (!SysHandleEvent(&event))
        {
            if (!MenuHandleEvent(0, &event, &error))
            {
                if (!ApplicationHandleEvent(&event))
                {
```

```
                FrmDispatchEvent(&event);
            }
        }
      }
    } while (event.eType != appStopEvent);
}




/**
 * Processa o evento frmLoadEvent recebido pela aplicação.
 *
 * @param event Evento
 * @return Verdadeiro se o evento foi interceptado, falso caso
contrário.
 */
static Boolean ApplicationHandleEvent(EventType *event)
{
      UInt16 formId;
      FormPtr frmP;
    UInt16 itemID;

      if (event->eType == frmLoadEvent)
      {
            /* carrega o resource do form */
            formId = event->data.frmLoad.formID;
            frmP = FrmInitForm(formId);
            FrmSetActiveForm(frmP);

            // Seta o EventHandler de cada form
            // event.
            switch (formId)
            {
                  case SelClienteForm:
                        FrmSetEventHandler(frmP,
SelClienteHandleEvent);
                        break;
                  case SelProdutoForm:
                        FrmSetEventHandler(frmP,
SelProdutoHandleEvent);
                        break;
                  case InfoProdutoForm:
                        FrmSetEventHandler(frmP, DefaultHandleEvent);
                        break;
                  case AdicionaPedidoDlg:
                        FrmSetEventHandler(frmP, DefaultHandleEvent);
                        break;
                  case VizPedidoForm:
                        FrmSetEventHandler(frmP, DefaultHandleEvent);
                        break;
                  default:
                ErrFatalDisplay("Invalid Form Load Event");
                        break;

            }
            return true;
      }

    if (event->eType == menuEvent)
    {
        itemID = event->data.menu.itemID;
```

```
        switch(itemID) {
            case MaMAtuaCliMnI:
            {
                ClienteDbPopuleDb();
                return true;
            }
            case MaMAtuaProMnI:
            {
                ProdutoDbPopuleDb();
                return true;
            }
        }
    }

      return false;
}
```

```
/**
 * Verifica se a versão da ROM é suportada.
 *
 * @param requiredVersion Versão mínima requerida
 * @param lauchFlags flags que indicam se interface gráfica da
aplicação
 * foi inicializada.
 * @return Zero se for compatível, senão um código de erro.
 *
 */
static Err RomVersionCompatible(UInt32 requiredVersion, UInt16
launchFlags)
{
      UInt32 romVersion;

      // See if we're on in minimum required version of the ROM or
later.
      FtrGet(sysFtrCreator, sysFtrNumROMVersion, &romVersion);
      if (romVersion < requiredVersion)
            {
            if ((launchFlags & (sysAppLaunchFlagNewGlobals |
sysAppLaunchFlagUIApp)) ==
                  (sysAppLaunchFlagNewGlobals |
sysAppLaunchFlagUIApp))
                  {
                  FrmAlert (RomIncompatibleAlert);

                  // Palm OS 1.0 will continuously relaunch this app
unless we switch to
                  // another safe one.
                  if (romVersion <= kPalmOS10Version)
                        {
                        AppLaunchWithCommand(sysFileCDefaultApp,
sysAppLaunchCmdNormalLaunch, NULL);
                        }
                  }

            return sysErrRomIncompatible;
            }

      return errNone;
```

```
}


/** Retorna um pornteiro para um objeto do form corrente
 *
 * @param formId id do form
 * @return void *
 */


Boolean DefaultHandleEvent(EventType* event)
{

    Boolean handled = false;
    FormPtr frmP;

    switch (event->eType)
    {
        case frmOpenEvent:
            frmP = FrmGetActiveForm();
            FrmDrawForm (frmP);
            handled = true;
            break;
        case ctlSelectEvent:
            FrmAlert(MainAlert);
            handled = true;
            break;
        default:
            break;

    }
    return handled;
}
```

vendas.h:

```
#ifndef _VENDAS_H
#define _VENDAS_H

#include "PalmOS.h"
#include "SelCliente.h"
#include "SelProduto.h"



/* modificar para ver se o Palm OS é versão 3.5 ou superior */
/* Define the minimum OS version we support (2.0 for now). */
#define ourMinVersion   sysMakeROMVersion(2,0,0,sysROMStageRelease,0)
#define kPalmOS10Version      sysMakeROMVersion
(1,0,0,sysROMStageRelease,0)


#define appFileCreator              'rgs4'      // register your own
at http://www.palmos.com/dev/creatorid/
#define appVersionNum               0x01
#define appPrefID                   0x00
#define appPrefVersionNum           0x01
```

```
/***********************************************************************
**
 *
 *    Estruturas internas  de manipulação dos dados de preferências...
 *
 ***********************************************************************
**/
typedef struct
      {
      UInt8 replaceme;
      } StarterPreferenceType;

typedef struct
      {
      UInt8 replaceme;
      } StarterAppInfoType;

typedef StarterAppInfoType* StarterAppInfoPtr;




/* Função chamada pelo Application Manager */
static UInt32 StarterPalmMain(UInt16, MemPtr, UInt16);

/* Inicia o programa */
static Err StartApp(void);

/* Termina a execução do programa */
static void StopApp(void);

/* Controla os eventos recepcionados pelo programa */
static void EventLoop(void);

/* Processa evento */
static Boolean ApplicationHandleEvent(EventType*);

/* verifica se a versão da ROM é suportada */
static Err RomVersionCompatible(UInt32, UInt16);


/* Verifica versão da rom */
static Err RomVersionCompatible(UInt32, UInt16);

/* método que retorna um ponteiro para um objeto gráfico com base no
seu ID */
static void * GetObjectPtr(UInt16);

/* Handle padrão */
Boolean DefaultHandleEvent(EventType* event);


#endif
```

vendas.rcp:

```
MENU ID MainMenu
BEGIN
    PULLDOWN "Dados"
        BEGIN
```

```
            MENUITEM "Atualiza Clientes" ID MaMAtuaCliMnI
            MENUITEM "Atualiza Produtos" ID MaMAtuaProMnI
            MENUITEM SEPARATOR
            MENUITEM "Envia pedidos" ID MAM_PEDIDO_MNI
        END
END
FORM ID SelClienteForm AT (0 0 160 160)
MENUID MainMenu
BEGIN
    TITLE "Cliente"
    POPUPTRIGGER "Todas" ID SCF_RAMO_POPUP AT (120 0 AUTO AUTO)
RIGHTANCHOR
    TABLE ID SCF_CLIENTE_TABLE AT (2 20 130 100) ROWS 3 COLUMNS 1
COLUMNWIDTHS 100
    SCROLLBAR ID SCF_CLIENTE_SCROLL AT (135 20 7 140)
    BUTTON "Info" ID SCFInfoBtn AT ( 5  145 AUTO AUTO) LEFTANCHOR
FRAME FONT 0
    BUTTON "Selecionar" ID SCFSelecionaBtn AT (PREVRIGHT+3 145 AUTO
AUTO) LEFTANCHOR FRAME FONT 0
    BUTTON "Sair" ID SCFSairBtn AT (PREVRIGHT+3 145 AUTO AUTO)
LEFTANCHOR FRAME FONT 0
END

FORM SelProdutoForm AT (0 0 160 160)
MENUID MainMenu
BEGIN
    TITLE "Cliente:"
    LABEL "0 Itens" ID SPFItensLbl AT (100 0)
    TABLE ID SPF_PRODUTO_TABLE AT (2 20 130 100) ROWS 3 COLUMNS 1
COLUMNWIDTHS 100
    BUTTON "Info" ID SPFInfoBtn AT (2  145 AUTO AUTO) LEFTANCHOR FRAME
FONT 0
    BUTTON "Adiciona" ID SPFAdicionaBtn AT (PREVRIGHT+3 145 AUTO AUTO)
LEFTANCHOR FRAME FONT 0
    BUTTON "Pedido" ID SPFPedidoBtn AT (PREVRIGHT+3 145 AUTO AUTO)
LEFTANCHOR FRAME FONT 0
    BUTTON "Sair" ID SPFSairBtn AT (PREVRIGHT+3 145 AUTO AUTO)
LEFTANCHOR FRAME FONT 0
END

FORM  InfoProdutoForm AT (2 2 156 156)
MODAL
BEGIN
    TITLE "Cliente:"
    LABEL "Descricao:" ID IPFDescricaoLbl AT (3 20)
    FIELD ID IPFDescricaoFld AT (PREVRIGHT PREVTOP 120 15) NONEDITABLE
UNDERLINED MULTIPLELINES MAXCHARS 40
    LABEL "Embalagem:" ID IPFEmbalagemLbl AT (3 PREVBOTTOM+2)
    FIELD ID IPFEmbalagemFld AT (PREVRIGHT PREVTOP 120 15) NONEDITABLE
UNDERLINED MAXCHARS 40
    LABEL "Estoque:" ID IPFEstoqueLbl AT (3 PREVBOTTOM+2)
    FIELD ID IPFEstoqueFld AT (PREVRIGHT PREVTOP 120 15) NONEDITABLE
UNDERLINED MAXCHARS 40
    LABEL "Preço:" ID IPFPrecoLbl AT (3 PREVBOTTOM+2)
    FIELD ID IPFPrecoFld AT (PREVRIGHT PREVTOP 120 15) NONEDITABLE
UNDERLINED MAXCHARS 40
    BUTTON "Adiciona Item" ID IPFAdicionaBtn AT (30 130 AUTO AUTO)
LEFTANCHOR FRAME FONT 0
    BUTTON "Volta" ID IPFVoltarBtn AT (PREVRIGHT+3 130 AUTO AUTO)
LEFTANCHOR FRAME FONT 0
END

FORM InfoClienteForm AT (2 2 156 156)
```

```
USABLE MODAL SAVEBEHIND
BEGIN
    TITLE "Cliente:"
    LABEL "Codigo:" ID ICF_CODIGO_LBL AT (5 20)
    FIELD ID ICF_CODIGO_FLD AT (PREVRIGHT PREVTOP 30 15) NONEDITABLE
UNDERLINED MAXCHARS 30
    LABEL "Nome:" ID ICF_NOME_LBL AT (3 PREVBOTTOM+2)
    FIELD ID ICF_NOME_FLD AT (PREVRIGHT PREVTOP 120 15) NONEDITABLE
UNDERLINED MULTIPLELINES MAXCHARS 100
    LABEL "Ramo:" ID ICF_RAMO_LBL AT (3 PREVBOTTOM+2)
    FIELD ID ICF_RAMO_FLD AT (PREVRIGHT PREVTOP 80 15) NONEDITABLE
UNDERLINED MULTIPLELINES MAXCHARS 100
    LABEL "Endereço:" ID ICFEnderecoLbl AT (3 PREVBOTTOM+2)
    FIELD ID ICF_ENDERECO_FLD AT (PREVRIGHT PREVTOP 120 15)
NONEDITABLE UNDERLINED MULTIPLELINES MAXCHARS 100
    LABEL "Contato:" ID ICFContatoLbl AT (3 PREVBOTTOM+2)
    FIELD ID ICF_CONTATO_FLD AT (PREVRIGHT PREVTOP 120 15) NONEDITABLE
UNDERLINED MULTIPLELINES MAXCHARS 100
    BUTTON "OK" ID ICFOKBtn AT (50 140 AUTO AUTO) LEFTANCHOR FRAME
FONT 0
END


FORM AdicionaPedidoDlg AT (10 10 130 130)
MODAL
BEGIN
    TITLE "Cliente"
    LABEL "Produto:" ID APDNomeLbl AT (3 20)
    FIELD ID APDDescricaoFld AT (PREVRIGHT PREVTOP AUTO AUTO)
NONEDITABLE MAXCHARS 80
    LABEL "Quantidade" ID APDQuantidadeLbl AT (3 PREVBOTTOM+2)
    FIELD ID APDQuantidadeFld AT (PREVRIGHT PREVTOP AUTO AUTO)
UNDERLINED MAXCHARS 15
    BUTTON "Adiciona" ID APDAdicionaBtn AT (20 110 AUTO AUTO)
LEFTANCHOR FRAME FONT 0
    BUTTON "Cancela"  ID APDCancelaBtn AT (PREVRIGHT+5 PREVTOP AUTO
AUTO) RIGHTANCHOR FRAME FONT 0
END


FORM VizPedidoForm AT (0 0 160 160)
MENUID MainMenu
BEGIN
    TITLE "Pedido"
    LABEL "Cliente:" ID VPFNomeLbl AT (3 20)
    FIELD ID VPFClienteFld AT (PREVRIGHT PREVTOP 80 15) NONEDITABLE
MAXCHARS 80
    LABEL "Lista de Produto:" ID VPFListaLbl AT (3 PREVBOTTOM+3)
    TABLE ID VPFProdutosTbe AT (0 PREVBOTTOM+3 90 140) ROWS 4 COLUMNS
3
    BUTTON "Envia" ID VPFEnviaBtn AT (20 145 AUTO AUTO) LEFTANCHOR
FRAME FONT 0
    BUTTON "Cancela"  ID VPFCancelaBtn AT (PREVRIGHT+5 PREVTOP AUTO
AUTO) RIGHTANCHOR FRAME FONT 0
END


ALERT ID RomIncompatibleAlert INFORMATION
BEGIN
    TITLE "Problema"
    MESSAGE "ROM Incompatível com a aplicação"
    BUTTONS "OK"
END
```

```
ALERT ID MainAlert INFORMATION
BEGIN
    TITLE "Um alerta !"
    MESSAGE "Você apertou o botão."
    BUTTONS "OK"
END

ALERT ID FormInvalidoAlert INFORMATION
BEGIN
    TITLE "Form Inválido !"
    MESSAGE "Form inválido !!!!"
    BUTTONS "OK"
END
```

vendasdb.c:

```
#include "VendasDbs.h"



/***********************************************************************
******
  Funções Internas do módulo
 ***********************************************************************
******/


 /* ClienteDb */

static UInt16 ClienteDbCompareRegistros(ClienteRecord* r1,
      ClienteRecord* r2, Int16 other,
      SortRecordInfoPtr info1, SortRecordInfoPtr info2,
      MemHandle appInfoH);



/* RamoDb */
static UInt16 RamoDbCompareRegistros(RamoRecord* r1,
      RamoRecord* r2, Int16 other,
      SortRecordInfoPtr info1, SortRecordInfoPtr info2,
      MemHandle appInfoH);




/* Ponteiro para os Databases */
DmOpenRef ClienteDbPtr = 0;
DmOpenRef RamoDbPtr = 0;
DmOpenRef ProdutoDbPtr = 0;







/***********************************************************************
*
  Funções de manipulação do ClienteDb
***********************************************************************
/
```

```
Err ClienteDbAbre(void)
{

      // declarations
      const UInt32 dbType = ClienteDBType;
      const UInt32 dbCreator = ClienteDBCreator;
      const char *dbName = ClienteDBName;
      const int dbCardNumber = 0;
    UInt16 version = 1;



    Err erro;
    LocalID dbID;
    DmOpenRef dbP;
    UInt16 cardNo;
    erro = errNone;

    if (ClienteDbPtr == 0 ) {
        /* Procura o ID do banco */
        dbP = DmOpenDatabaseByTypeCreator(dbType, dbCreator,
dmModeReadWrite);

        if (dbP == 0) {

            /* Não foi achado, então criar banco de dados */
            erro = DmCreateDatabase(dbCardNumber, dbName, dbCreator,
                                    dbType, false);
            if (erro != errNone) {
                return erro;
            }

            dbID = DmFindDatabase(0, ClienteDBName);
            if (dbID == 0 ) {
                return DmGetLastErr();
            }

            dbP = DmOpenDatabase(0, dbID, dmModeReadWrite);
            if ( 0 ==dbP) {
                return DmGetLastErr();
            }
            DmSetDatabaseInfo(0, dbID, NULL, NULL, &version, NULL,
NULL, NULL, NULL, NULL, NULL, NULL, NULL);
        }
        ClienteDbPtr = dbP;
    }
    return erro;
}

static UInt16 ClienteDbCompareRegistros(ClienteRecord* r1,
      ClienteRecord* r2, Int16 other,
      SortRecordInfoPtr info1, SortRecordInfoPtr info2,
      MemHandle appInfoH)
{

    if (other == 0) {
        /* comparar pelo nome do cliente */
        return StrCompare(r1->nome, r2->nome);
    } else {
        if (r1->codigo < r2->codigo) return -1;
        if (r1->codigo > r2->codigo) return 1;
```

```
        return 0;
    }
}


Err ClienteDbFecha(void)
{
    Err erro;

    if (ClienteDbPtr !=0)
    {
        erro == DmCloseDatabase(ClienteDbPtr);
        return erro;
    }
    return errNone;
}


Err ClienteDbAdiciona(UInt16 codigo, UInt16 ramo,  Char* nome)
{
    Err erro = 0;
    Char msgErro[100];
    UInt16 index;
    UInt16 nomeSize;
    MemHandle newRecord;
    MemPtr newRecordPtr;
    ClienteRecord cliente;


    MemSet(&cliente, 0, sizeof(ClienteRecord));
    cliente.codigo = codigo;
    cliente.ramo = ramo;
    nomeSize = StrLen(nome);

    if (nomeSize > CLIENTE_NOME_SIZE - 1) {
        nomeSize = CLIENTE_NOME_SIZE - 1;
    }
    StrNCopy(cliente.nome, nome, nomeSize);
    cliente.nome[nomeSize] = '\0';

    index = DmFindSortPositionV10(ClienteDbPtr, &cliente, (DmComparF
*)ClienteDbCompareRegistros, 0);

    newRecord = DmNewRecord(ClienteDbPtr, &index, sizeof
(ClienteRecord));
    newRecordPtr = MemHandleLock(newRecord);
    DmSet(newRecordPtr, 0, sizeof(ClienteRecord), 0);
    erro = DmWrite(newRecordPtr, 0, &cliente, sizeof(ClienteRecord));
    if (erro!=errNone) {
        SysFatalAlert("Registro não foi gravado");
        SysFatalAlert(SysErrString(erro, msgErro, 100));
    }

    MemHandleUnlock(newRecord);
    DmReleaseRecord(ClienteDbPtr, index, false);

    return erro;
}



void ClienteDbPopuleDb()
```

```
{
    Char *nome1 = "Clube 12 de Agosto";
    Char *nome2 = "Angeloni Supermercado";
    Char *nome3 = "João de Barro";
    UInt16 ramo1 = 1;
    UInt16 ramo2 = 2;
    UInt16 ramo3 = 3;
    UInt16 codigo1 = 1;
    UInt16 codigo2 = 2;
    UInt16 codigo3 = 3;
    Err erro = errNone;

    erro = ClienteDbAdiciona(codigo1,ramo1, nome1);
    erro = ClienteDbAdiciona(codigo2,ramo2, nome2);
    erro = ClienteDbAdiciona(codigo3,ramo3, nome3);
}




/*====================================================================
*/



/*********************************************************************
  Funções de manipulação do RamoDb
 *********************************************************************
/


Err RamoDbAbre(void)
{

      // declarations
      const UInt32 dbType = RamoDbType;
      const UInt32 dbCreator = RamoDbCreator;
      const Char *dbName = RamoDbName;
      const UInt16 dbCardNumber = 0;
    UInt16 version = 1;


    Err erro;
    LocalID dbID;
    DmOpenRef dbP;
    erro = errNone;

    if (RamoDbPtr==0) {

        /* Procura o ID do banco */
        dbP = DmOpenDatabaseByTypeCreator(RamoDbType, RamoDbCreator,
dmModeReadWrite);

        if (dbP == 0) {
            /* Não foi achado, então criar banco de dados */
            erro = DmCreateDatabase(dbCardNumber, dbName, dbCreator,
                                    dbType, false);
            if (erro != errNone) {
                return erro;
            }
            dbID = DmFindDatabase(0, RamoDbName);
```

```
            if (dbID == 0 ) {
                return DmGetLastErr();
            }

            dbP = DmOpenDatabase(0, dbID, dmModeReadWrite);
            if ( 0 == dbP) {
                return DmGetLastErr();
            }
            DmSetDatabaseInfo(0, dbID, NULL, NULL, &version, NULL,
NULL, NULL, NULL, NULL, NULL, NULL, NULL);
          }
          RamoDbPtr = dbP;
    }
    return erro;
}



Err RamoDbFecha(void)
{
    Err erro;
    erro == DmCloseDatabase(RamoDbPtr);
    if (erro == errNone) {
        RamoDbPtr = 0;
        return errNone;
    }
    return erro;
}


static UInt16 RamoDbCompareRegistros(RamoRecord* r1,
      RamoRecord* r2, Int16 other,
      SortRecordInfoPtr info1, SortRecordInfoPtr info2,
      MemHandle appInfoH)
{

    if (other == 0) {
        /* comparar pelo nome do cliente */
        return StrCompare(r1->nome, r2->nome);
    } else {
        if (r1->codigo < r2->codigo) return -1;
        if (r1->codigo > r2->codigo) return 1;
        return 0;
    }
}



Err RamoDbAdiciona(UInt16 codigo, Char* nome)
{
    Err erro = 0;
    UInt16 index;
    MemHandle newRecord;
    MemPtr newRecordPtr;
    RamoRecord *ramo;

    ramo = MemPtrNew(sizeof(RamoRecord));
    ramo->codigo = codigo;
    StrNCopy(ramo->nome, nome, StrLen(nome)-1);
    ramo->nome[StrLen(nome)-1] = chrNull;
```

```
    index = DmFindSortPositionV10(RamoDbPtr, &ramo, (DmComparF *)
RamoDbCompareRegistros, 0);

    newRecord = DmNewRecord(RamoDbPtr, &index, sizeof(RamoRecord));
    newRecordPtr = MemHandleLock(newRecord);

    erro = DmWrite(newRecordPtr, 0, ramo, sizeof(RamoRecord));

    ErrNonFatalDisplayIf(erro, "RamoDbAdiciona: Falha ao escrever o
novo ramo no registro");

    MemHandleUnlock(newRecord);
    DmReleaseRecord(RamoDbPtr, index, false);


    return erro;
}



void RamoDbPopuleDb()
{
    char *nome1 = "Clube";
    char *nome2 = "Supermercado";
    char *nome3 = "Restaurante";
    int codigo1 = 1;
    int codigo2 = 2;
    int codigo3 = 3;
    Err erro = errNone;

    erro = RamoDbAbre();
    ErrNonFatalDisplayIf(erro, "RamoDbPopuleDb: Falha ao abrir o banco
de dados");
    if (erro) return;
    erro = RamoDbAdiciona(codigo1, nome1);
    ErrNonFatalDisplayIf(erro, "RamoDbPopuleDb: Falha ao adicionar
dado");
    erro = RamoDbAdiciona(codigo2, nome2);
    ErrNonFatalDisplayIf(erro, "RamoDbPopuleDb: Falha ao adicionar
dado");
    erro = RamoDbAdiciona(codigo3, nome3);
    ErrNonFatalDisplayIf(erro, "RamoDbPopuleDb: Falha ao adicionar
dado");
}


/*==========================================================================
===*/




/*************************************************************************
****
  Funções de manipulação da tabela de Produtos
 *************************************************************************
****/


Err ProdutoDbAbre(void)
{
```

```
      // declarations
      const UInt32 dbType = ProdutoDbType;
      const UInt32 dbCreator = ProdutoDbCreator;
      const char *dbName = ProdutoDbName;
      const int dbCardNumber = 0;
    UInt16 version = 1;



    Err erro;
    LocalID dbID;
    DmOpenRef dbP;
    UInt16 cardNo;
    erro = errNone;

    if (ProdutoDbPtr == 0 ) {
        /* Procura o ID do banco */
        dbP = DmOpenDatabaseByTypeCreator(dbType, dbCreator,
dmModeReadWrite);

        if (dbP == 0) {

            /* Não foi achado, então criar banco de dados */
            erro = DmCreateDatabase(dbCardNumber, dbName, dbCreator,
                                    dbType, false);
            if (erro != errNone) {
                return erro;
            }

            dbID = DmFindDatabase(0, ProdutoDbName);
            if (dbID == 0 ) {
                return DmGetLastErr();
            }

            dbP = DmOpenDatabase(0, dbID, dmModeReadWrite);
            if ( 0 ==dbP) {
                return DmGetLastErr();
            }
            DmSetDatabaseInfo(0, dbID, NULL, NULL, &version, NULL,
NULL, NULL, NULL, NULL, NULL, NULL, NULL);
        }
        ProdutoDbPtr = dbP;
    }
    return erro;
}

static UInt16 ProdutoDbCompareRegistros(ProdutoRecord* r1,
      ProdutoRecord* r2, Int16 other,
      SortRecordInfoPtr info1, SortRecordInfoPtr info2,
      MemHandle appInfoH)
{

    if (other == 0) {
        /* comparar pelo nome do cliente */
        return StrCompare(r1->descricao, r2->descricao);
    } else {
        if (r1->codigo < r2->codigo) return -1;
        if (r1->codigo > r2->codigo) return 1;
        return 0;
    }
}
```

```
Err ProdutoDbFecha(void)
{
    Err erro;

    if (ClienteDbPtr !=0)
    {
        erro == DmCloseDatabase(ProdutoDbPtr);
        return erro;
    }
    return errNone;
}


Err ProdutoDbAdiciona(UInt16 codigo, UInt16 tipo,  Char* descricao)
{
    Err erro = 0;
    Char msgErro[100];
    UInt16 index;
    MemHandle newRecord;
    MemPtr newRecordPtr;
    ProdutoRecord produto;


    MemSet(&produto, 0, sizeof(ProdutoRecord));
    produto.codigo = codigo;
    produto.tipo = tipo;

    StrNCopy(produto.descricao, descricao, PRODUTO_DESCRICAO_SIZE-1);
    produto.descricao[PRODUTO_DESCRICAO_SIZE-1] = '\0';

    index = DmFindSortPositionV10(ProdutoDbPtr, &produto, (DmComparF
*)ProdutoDbCompareRegistros, 0);

    newRecord = DmNewRecord(ProdutoDbPtr, &index, sizeof
(ProdutoRecord));
    newRecordPtr = MemHandleLock(newRecord);
    DmSet(newRecordPtr, 0, sizeof(ProdutoRecord), 0);
    erro = DmWrite(newRecordPtr, 0, &produto, sizeof(ProdutoRecord));
    if (erro!=errNone) {
        SysFatalAlert("Registro não foi gravado");
        SysFatalAlert(SysErrString(erro, msgErro, 100));
    }

    MemHandleUnlock(newRecord);
    DmReleaseRecord(ProdutoDbPtr, index, false);

    return erro;
}



void ProdutoDbPopuleDb()
{
    Char *nome1 = "Picanha";
    Char *nome2 = "Maminha";
    Char *nome3 = "Cerveja";
    UInt16 tipo1 = 1;
    UInt16 codigo1 = 1;
    UInt16 codigo2 = 2;
    UInt16 codigo3 = 3;
    Err erro = errNone;
```

```
    erro = ProdutoDbAdiciona(codigo1,tipo1, nome1);
    erro = ProdutoDbAdiciona(codigo2,tipo1, nome2);
    erro = ProdutoDbAdiciona(codigo3,tipo1, nome3);
}

/*=================================================================*/
```

vendasdb.h:

```
#ifndef _VENDASDBS_H_
#define _VENDASDBS_H_

#include "PalmOS.h"
#include "DataMgr.h"
#include "VendasTypes.h"



#define ClienteDBName "Vendas_Cliente"
#define ClienteDBType 'CLIE'
#define ClienteDBCreator 'rgs4'

struct StructClienteRecord {
    UInt16 codigo;
    UInt16 ramo;
    Char nome[CLIENTE_NOME_SIZE];
};

typedef struct StructClienteRecord ClienteRecord;

Err ClienteDbAbre(void);
Err ClienteDbFecha(void);
Err ClienteDbAdiciona(UInt16, UInt16, Char*);
void ClienteDbPopuleDb(void);




#define RamoDbName "Vendas_Ramo"
#define RamoDbType 'RAMO'
#define RamoDbCreator 'rgs4'
#define RamoNomeMaxSize 60

struct StructRamoRecord {
    UInt16 codigo;
    Char nome[RamoNomeMaxSize];
};

typedef struct StructRamoRecord RamoRecord;

Err RamoDbAbre(void);
Err RamoDbFecha(void);
Err RamoDbAdiciona(UInt16, Char*);
void RamoDbPopuleDb(void);



#define ProdutoDbName "Vendas_Produto"
#define ProdutoDbType 'PROD'
#define ProdutoDbCreator 'rgs4'
```

```
struct StructProdutoRecord {
    UInt16 codigo;
    UInt16 tipo;
    Char descricao[PRODUTO_DESCRICAO_SIZE];
};

typedef struct StructProdutoRecord ProdutoRecord;

Err ProdutoDbAbre(void);
Err ProdutoDbFecha(void);
Err ProdutoDbAdiciona(UInt16, UInt16, Char*);
void ProdutoDbPopuleDb(void);



#endif
```

vendasprc.h:

```
/* pilrc generated file.  Do not edit!*/
#define FormInvalidoAlert 9942
#define MainAlert 9943
#define RomIncompatibleAlert 9944
#define VPFCancelaBtn 9945
#define VPFEnviaBtn 9946
#define VPFProdutosTbe 9947
#define VPFListaLbl 9948
#define VPFClienteFld 9949
#define VPFNomeLbl 9950
#define VizPedidoForm 9951
#define APDCancelaBtn 9952
#define APDAdicionaBtn 9953
#define APDQuantidadeFld 9954
#define APDQuantidadeLbl 9955
#define APDDescricaoFld 9956
#define APDNomeLbl 9957
#define AdicionaPedidoDlg 9958
#define ICFOKBtn 9959
#define ICF_CONTATO_FLD 9960
#define ICFContatoLbl 9961
#define ICF_ENDERECO_FLD 9962
#define ICFEnderecoLbl 9963
#define ICF_RAMO_FLD 9964
#define ICF_RAMO_LBL 9965
#define ICF_NOME_FLD 9966
#define ICF_NOME_LBL 9967
#define ICF_CODIGO_FLD 9968
#define ICF_CODIGO_LBL 9969
#define InfoClienteForm 9970
#define IPFVoltarBtn 9971
#define IPFAdicionaBtn 9972
#define IPFPrecoFld 9973
#define IPFPrecoLbl 9974
#define IPFEstoqueFld 9975
#define IPFEstoqueLbl 9976
#define IPFEmbalagemFld 9977
#define IPFEmbalagemLbl 9978
#define IPFDescricaoFld 9979
#define IPFDescricaoLbl 9980
#define InfoProdutoForm 9981
#define SPFSairBtn 9982
```

```
#define SPFPedidoBtn 9983
#define SPFAdicionaBtn 9984
#define SPFInfoBtn 9985
#define SPF_PRODUTO_TABLE 9986
#define SPFItensLbl 9987
#define SelProdutoForm 9988
#define SCFSairBtn 9989
#define SCFSelecionaBtn 9990
#define SCFInfoBtn 9991
#define SCF_CLIENTE_SCROLL 9992
#define SCF_CLIENTE_TABLE 9993
#define SCF_RAMO_POPUP 9994
#define SelClienteForm 9995
#define MAM_PEDIDO_MNI 9996
#define MaMAtuaProMnI 9997
#define MaMAtuaCliMnI 9998
#define MainMenu 9999
```

vendastypes.h:

```
/*********************************************************************
*****
 * Arquivo contendo definições de estruturas de dados utilizadas
 * em todo o sistema.
 *
 *
 *
 */



#ifndef _VENDASTYPES_H_
#define _VENDASTYPES_H_



/* tamanho dos campos de dados de ramo:
   Sempre somados em 1, pois em C as strings sempre devem terminar
   em caracter nulo ;-) */


#define RAMO_NOME_SIZE 101

typedef struct {
    UInt16 ramoId;
    Char nome[RAMO_NOME_SIZE];
} RamoType;




/* tamanho dos campos de dados do cliente:
   Sempre somados em 1, pois em C as strings sempre devem terminar
   em caracter nulo ;-) */

#define CLIENTE_NOME_SIZE 101
#define CLIENTE_RESPONSAVEL_SIZE 101
#define CLIENTE_ENDERECO_SIZE 101
#define CLIENTE_BAIRRO_SIZE 101
#define CLIENTE_CIDADE_SIZE 101
#define CLIENTE_ESTADO_SIZE 3
#define CLIENTE_EMAIL_SIZE 101
```

```
typedef struct {
    UInt16 clienteId;
    UInt16 ramoId;
    UInt16 vendedorId;
    Char nome[CLIENTE_NOME_SIZE];
    Char responsavel[CLIENTE_RESPONSAVEL_SIZE];
    Char aniversario[11];
    Char endereco[CLIENTE_ENDERECO_SIZE];
    Char bairro[CLIENTE_BAIRRO_SIZE];
    Char cidade[CLIENTE_CIDADE_SIZE];
    Char estado[CLIENTE_ESTADO_SIZE];
    Char email[CLIENTE_EMAIL_SIZE];
} ClienteType;



/* tamanho dos campos de dados do produto: lembrando que as strings
   devem terminar em nulo ... */
#define TIPO_SIZE 101

typedef struct {
    UInt16 tipoId;
    Char tipo[TIPO_SIZE];
} TipoType;


/* tamanho dos campos de dados do produto: lembrando que as strings
   devem terminar em nulo ... */

#define PRODUTO_DESCRICAO_SIZE 101
#define PRODUTO_EMBALAGEM_SIZE 51

typedef struct {
    UInt16 produtoId;
    UInt16 tipoId;
    Char descricao[PRODUTO_DESCRICAO_SIZE];
    Char embalagem[PRODUTO_EMBALAGEM_SIZE];
    float preco;
    UInt16 estoque;
    UInt16 quantidade;
 } ProdutoType;



#endif
```

wsdl.c:

```
#include "soapH.h"
#include "soapServicoService.nsmap"
main()
{
      struct soap soap;
      soap_init(&soap);

      if (soap_call_tns__addProdutoPedido ( &soap,
"http://grumiche.ath.cx:8080/service/servidor.php",
"urn:Servico#servico#addProdutoPedido",/* xsd__int  pedidoID, xsd__int
produtoID, xsd__int  quantidade, struct tns__addProdutoPedidoResponse
* out*/ ))
```

```
            soap_print_fault(&soap,stderr);


      if (soap_call_tns__getInfoProduto ( &soap,
"http://grumiche.ath.cx:8080/service/servidor.php",
"urn:Servico#servico#getInfoProduto",/* xsd__int  produtoID, struct
tns__getInfoProdutoResponse * out*/ ))
            soap_print_fault(&soap,stderr);


      if (soap_call_tns__getPedido ( &soap,
"http://grumiche.ath.cx:8080/service/servidor.php",
"urn:Servico#servico#getPedido",/* xsd__int  clienteID, xsd__int
vendedorID, xsd__string  senha, struct tns__getPedidoResponse *
out*/ ))
            soap_print_fault(&soap,stderr);


      if (soap_call_tns__getEstoque ( &soap,
"http://grumiche.ath.cx:8080/service/servidor.php",
"urn:Servico#servico#getEstoque",/* xsd__int  produtoID, struct
tns__getEstoqueResponse * out*/ ))
            soap_print_fault(&soap,stderr);


      if (soap_call_tns__getInfoCliente ( &soap,
"http://grumiche.ath.cx:8080/service/servidor.php",
"urn:Servico#servico#getInfoCliente",/* xsd__int  clienteID, struct
tns__getInfoClienteResponse * out*/ ))
            soap_print_fault(&soap,stderr);


      if (soap_call_tns__getComissao ( &soap,
"http://grumiche.ath.cx:8080/service/servidor.php",
"urn:Servico#servico#getComissao",/* xsd__int  vendedorID, xsd__string
senha, struct tns__getComissaoResponse * out*/ ))
            soap_print_fault(&soap,stderr);


}
```

Servidor

Cliente.php:

```
<?php
require_once 'MySQL.php';

class Cliente {
    var $_mysql;
    var $_existe;
      var $_dados = array();

    function Cliente($clienteID = 0)
    {
        $this->_mysql = new MySQL();
        if ($clienteID != 0) {
            $this->_get($clienteID);
        }
    }
```

```
    function _get($clienteID)
    {
        $get = "
SELECT clienteID, nome, endereco, bairro, cidade, estado
FROM Cliente
WHERE clienteID = $clienteID";
        $resultado = $this->_mysql->getAll($get);
        if (empty($resultado)) {
            $this->_existe = FALSE;
        } else {
                  $this->_dados = $resultado[0];
            $this->_existe = TRUE;
        }
    }

      function get()
      {
            return $this->_dados;
      }

    function existe()
    {
        return $this->_existe;
    }

    function getLista()
    {
        $getLista = '
SELECT clienteID, nome
FROM Cliente
ORDER BY nome';
        return $this->_mysql->getAll($getLista);
    }
}
?>
```

config.php:

```
<?php
$host = 'localhost';
$banco = 'tcc';
$login = 'antonio';
$senha = 'tcc';
?>
```

MySQL.php:

```
<?php
require_once 'DB.php';
require_once 'config.php';

class MySQL {
      var $_db;
      var $_host = 'localhost';
      var $_banco = 'tcc';
      var $_login = 'antonio';
      var $_senha = 'tcc';

      var $_resultado;
```

```
	function MySQL()
	{
		$dsn = sprintf ("mysql://%s:%s@%s/%s", $this->_login,
$this->_senha, $this->_host, $this->_banco);
		$this->_db =& DB::connect($dsn, FALSE);
		if (DB::isError($this->_db)) {
			die($this->_db->getNessage());
		}
	}

	function _isError()
	{
		if (DB::isError($this->_resultado)) {
			return FALSE;
		}
	}

	function getAssoc($query, $force_array = FALSE, $params = array
(), $fetchmode = DB_FETCHMODE_ASSOC, $group = TRUE)
	{
		$this->_resultado = $this->_db->getAssoc($query,
$force_array, $params, $fetchmode, $group);
		$this->_isError();

		return $this->_resultado;
	}

	function getCol($query, $col = 0, $params = array())
	{
		$this->_resultado = $this->_db->getCol($query, $col,
$params);
		$this->_isError();

		return $this->_resultado;
	}

	function getOne($query, $params = array())
	{
		$this->_resultado = $this->_db->getOne($query, $params);
		$this->_isError();

		return $this->_resultado;
	}

	function getAll($query, $params = array(), $fetchmode =
DB_FETCHMODE_ASSOC)
	{
		$this->_resultado = $this->_db->getAll($query, $params,
$fetchmode);
		$this->_isError();

		return $this->_resultado;
	}

	function query($query, $params = array())
	{
		$this->_resultado = $this->_db->query($query, $params);
		$this->_isError();

		return $this->_resultado;
	}
}
?>
```

Pedido.php:

```
<?php
require_once 'MySQL.php';
require_once 'Cliente.php';
require_once 'Vendedor.php';

class Pedido {
      var $_mysql;
      var $_erro = FALSE;
      var $_errno = 0;

      function Pedido($clienteID = 0, $vendedorID = 0, $senha = '',
$init = FALSE)
      {
            $this->_mysql = new MySQL();

            if (!$init) {
                return TRUE;
            }

            $cliente = new Cliente($clienteID);
            $vendedor = new Vendedor($vendedorID, $senha);

            if (!$cliente->existe()) {
                  $this->_erro = TRUE;
                  $this->_errno = -1;
            } elseif (!$vendedor->existe()) {
                  $this->_erro = TRUE;
                  $this->_errno = -2;
            } elseif (!$vendedor->verifique($clienteID)) {
                  $this->_erro = TRUE;
                  $this->_errno = -3;
            }

            if (!$this->_erro) {
                  $this->_set($clienteID);
            }
      }

      function _set($clienteID)
      {
            $set = sprintf('
INSERT INTO
Pedido (clienteID, data)
VALUES (%s, NOW())', $clienteID);
            $this->_mysql->query($set);
            $lastID = 'SELECT LAST_INSERT_ID() AS pedidoID FROM
Pedido';
            $this->_pedidoID = $this->_mysql->getOne($lastID);
      }

      function getPedido()
      {
            return $this->_pedidoID;
      }

      function getErro()
      {
            return $this->_erro;
      }
```

```
      function getNumeroErro()
      {
            return $this->_errno;
      }

      function addProduto($pedidoID, $produtoID, $quantidade)
      {
            $addProduto = sprintf('
INSERT INTO
PedidoProduto (pedidoID, produtoID, quantidade)
VALUES (%s, %s, %s)', $pedidoID, $produtoID, $quantidade);
            $resultado = $this->_mysql->query($addProduto);
            if (!DB::isError($resultado)) {
                  $this->_updateProduto($produtoID, $quantidade);
                  return TRUE;
            } else {
                  return FALSE;
            }
      }

      function _updateProduto($produtoID, $quantidade)
      {
            $updateProduto = sprintf('
UPDATE Produto
SET quantidade = quantidade - %d
WHERE produtoID = %d', $quantidade, $produtoID);
            $this->_mysql->query($updateProduto);
      }
}
?>
```

Produto.php:

```
<?php
require_once 'MySQL.php';

class Produto {
    var $_mysql;
    var $_existe;
    var $_dados = array();

    function Produto($produtoID = 0)
    {
        $this->_mysql = new MySQL();
        if ($produtoID != 0) {
            $this->_get($produtoID);
        }
    }

    function _get($produtoID)
    {
        $get = "
SELECT produtoID, tipoID, descricao, embalagem, preco, quantidade
FROM Produto
WHERE produtoID = $produtoID";
        $resultado = $this->_mysql->getAll($get);
        if (empty($resultado)) {
            $this->_existe = FALSE;
        } else {
            $this->_existe = TRUE;
            $this->_dados = $resultado[0];
        }
```

```
    }

      function existe()
      {
            return $this->_existe;
      }

    function getLista()
    {
        $getLista = '
SELECT produtoID, tipoID, descricao, preco, quantidade
FROM Produto
ORDER BY descricao
LIMIT 0, 20';
        return $this->_mysql->getAll($getLista);
    }

    function get()
    {
        return $this->_dados;
    }

    function getEstoque()
    {
        if ($this->_existe) {
            $qtd = (int) $this->_dados['quantidade'];
        } else {
            $qtd = -1;
      }
      return $qtd;
      }
}
?>
```

Servico.php:

```
<?php
require_once 'SOAP/Value.php';

class Servico {
      var $__dispatch_map = array();
      var $__typedef = array();

      function Servico()
      {
            $this->__typedef['InfoProduto'] =
                              array(
                                    'produtoID' => 'int',
                                    'descricao' => 'string',
                                    'preco' => 'float',
                                    'estoque' => 'int'
                              );

            $this->__typedef['InfoCliente'] =
                              array(
                                    'clienteID' => 'int',
                                    'nome' => 'string',
                                    'endereco' => 'string',
                                    'bairro' => 'string',
                                    'cidade' => 'string',
                                    'estado' => 'string'
                              );
```

```
            $this->__dispatch_map['getInfoProduto'] =
                                array(
                                      'in' => array('produtoID' =>
'int'),
                                      'out' => array('infoProduto' =>
'{urn:Servico}InfoProduto')
                                );

            $this->__dispatch_map['getInfoCliente'] =
                                array(
                                      'in' => array('clienteID' =>
'int'),
                                      'out' => array('infoCliente' =>
'{urn:Servico}InfoCliente')
                                );

            $this->__dispatch_map['getComissao'] =
                                array(
                                      'in' => array('vendedorID' =>
'int', 'senha' => 'string'),
                                      'out' => array('comissao' =>
'float')
                                );


            $this->__dispatch_map['getEstoque'] =
                                array(
                                      'in' => array('produtoID' =>
'int'),
                                      'out' => array('quantidade' =>
'int')
                                );

            $this->__dispatch_map['getPedido'] =
                                array(
                                      'in' => array(
                                                  'clienteID' =>
'int',
                                                  'vendedorID' =>
'int',
                                                  'senha' =>
'string',
                                                ),
                                      'out' => array('pedidoID' =>
'int')
                                );

            $this->__dispatch_map['addProdutoPedido'] =
                                array(
                                      'in' => array(
                                                  'pedidoID' =>
'int',
                                                  'produtoID' =>
'int',
                                                  'quantidade' =>
'int',
                                                ),
                                      'out' => array('adicionado' =>
'boolean')
                                );

      }
```

```
	function __dispatch($methodname)
	{
		if (isset($this->__dispatch_map[$methodname])) {
			return $this->__dispatch_map[$methodname];
		}
		return NULL;
	}

	function &getEstoque($produtoID)
	{
		require_once 'Produto.php';
		$produto = new Produto($produtoID);
		if ($produto->existe()) {
			return new SOAP_Value('quantidade', 'int', $produto-
>getEstoque());
		} else {
			return new SOAP_Value('quantidade', 'int', -1);
		}
	}

	function &getComissao($vendedorID, $pass)
	{
		require_once 'Vendedor.php';
		$vendedor = new Vendedor($vendedorID, $pass);
		if ($vendedor->existe()) {
			return new SOAP_Value('comissao', 'float',
$vendedor->getComissao());
		} else {
			return new SOAP_Value('comissao', 'int', -1);
		}
	}

	function &getInfoProduto($produtoID)
	{
		require_once 'Produto.php';
		$produto = new Produto($produtoID);
		if ($produto->existe()) {
			$produtos = $produto->get();
			$infoProduto['produtoID'] = new SOAP_Value
('produtoID', 'int', (int) $produtos['produtoID']);
			$infoProduto['descricao'] = new SOAP_Value
('descricao', 'string', $produtos['descricao']);
			$infoProduto['preco'] = new SOAP_Value('preco',
'float', (float) $produtos['preco']);
			$infoProduto['estoque'] = new SOAP_Value('estoque',
'int', (int) $produtos['quantidade']);

			return new SOAP_Value('infoProduto', '{urn:Servico}
InfoProduto', $infoProduto);
		} else {
			return new SOAP_Value('return', 'int', -1);
		}
	}

	function &getInfoCliente($clienteID)
	{
		require_once 'Cliente.php';
		$cliente = new Cliente($clienteID);
		if ($cliente->existe()) {
			$clientes = $cliente->get();
			$infoCliente['clienteID'] = new SOAP_Value
('clienteID', 'int', (int) $clientes['clienteID']);
```

```
                  $infoCliente['nome'] = new SOAP_Value('nome',
'string', $clientes['nome']);
                  $infoCliente['endereco'] = new SOAP_Value
('endereco', 'string', $clientes['endereco']);
                  $infoCliente['bairro'] = new SOAP_Value('bairro',
'string', $clientes['bairro']);
                  $infoCliente['cidade'] = new SOAP_Value('cidade',
'string', $clientes['cidade']);
                  $infoCliente['estado'] = new SOAP_Value('estado',
'string', $clientes['estado']);

                  return new SOAP_Value('infoCliente', '{urn:Servico}
InfoCliente', $infoCliente);
            } else {
                  return new SOAP_Value('return', 'int', -1);
            }
      }

      function &getPedido($clienteID, $vendedorID, $pass)
      {
            require_once 'Pedido.php';
            $pedido = new Pedido($clienteID, $vendedorID, $pass,
TRUE);
            if (!$pedido->getErro()) {
                  return new SOAP_Value('pedidoID', 'int', $pedido-
>getPedido());
            } else {
                  return new SOAP_Value('pedidoID', 'int', $pedido-
>getNumeroErro());
            }
      }

      function &addProdutoPedido($pedidoID, $produtoID, $quantidade)
      {
            require_once 'Pedido.php';
            $pedido = new Pedido();
          if ($pedido->addProduto($pedidoID, $produtoID, $quantidade))
{
                  return new SOAP_Value('adicionado', 'boolean',
(bool) TRUE);
            } else {
                  return new SOAP_Value('adicionado', 'boolean',
(bool) FALSE);
            }
      }
}
?>
```

servidor.php:

```
<?php
require_once 'SOAP/Server.php';
require_once 'Servico.php';

$options = array('use' => 'literal', 'style' => 'document');
$servidor = new SOAP_Server($options);
$servidor->_auto_translation = TRUE;
$servico = new Servico();

// adicionar nosso objeto ao servidor SOAP (namespace)
$servidor->addObjectMap($servico, 'urn:Servico');
```

```
if (isset($_SERVER['REQUEST_METHOD']) && $_SERVER['REQUEST_METHOD'] ==
'POST') {
      $servidor->service($HTTP_RAW_POST_DATA);
} else {
      require_once 'SOAP/Disco.php';
      $disco = new SOAP_DISCO_Server($servidor, 'Servico');
      header("Content-type: text/xml");
      if (isset($_SERVER['QUERY_STRING']) && strcasecmp($_SERVER
['QUERY_STRING'],'wsdl')==0) {
            echo $disco->getWSDL();
      } else {
            echo $disco->getDISCO();
      }
      exit;
}
?>
```

Vendedor.php:

```
<?php
require_once 'MySQL.php';

class Vendedor {
      var $_mysql;
      var $_existe;
      var $_vendedorID;

      function Vendedor($vendedorID = 0, $senha = '')
      {
            $this->_mysql = new MySQL();
            $this->_vendedorID = $vendedorID;
            if ($vendedorID != 0) {
                  $this->_get($vendedorID, $senha);
            }
      }

      function _get($vendedorID, $senha)
      {
            $get = "
SELECT vendedorID
FROM Vendedor
WHERE vendedorID = $vendedorID
AND senha = '$senha'";
            $this->_vendedorID = $this->_mysql->getOne($get);
            if (is_null($this->_vendedorID)) {
                  $this->_existe = FALSE;
            } else {
                  $this->_existe = TRUE;
            }
      }

      function verifique($clienteID)
      {
            $verifique = "
SELECT vendedorID
FROM Cliente
WHERE clienteID = $clienteID";
            if ($this->_vendedorID == $this->_mysql->getOne
($verifique)) {
                  return TRUE;
            } else {
                  return FALSE;
```

```
            }
      }

      function existe()
      {
            return $this->_existe;
      }

      function getComissao()
      {
            $getComissao = sprintf('
SELECT SUM(PedidoProduto.quantidade*Produto.preco)*0.1 AS comissao
FROM Cliente, Pedido, PedidoProduto, Produto
WHERE Pedido.pedidoID = PedidoProduto.pedidoID
AND PedidoProduto.produtoID = Produto.produtoID
AND Pedido.clienteID = Cliente.clienteID
AND Cliente.vendedorID = %s', $this->_vendedorID);
            return (float) $this->_mysql->getOne($getComissao);
      }
}

?>
```

REFERÊNCIAS BIBLIOGRÁFICAS

1. DANTAS, Mario. **Tecnologias de Rede de Comunicação e Computadores**. Rio de Janeiro, RJ: Axcel Books, 2002
2. NASDAQ. **The NASDAQ Stock Market**. Disponível em: <http://www.nasdaq.com>. Acesso em: 29 nov. 2003.
3. BABELFISH. **BabelFish: The Language Plataform for a Global Generation**. Disponível em: <http://www.babelfish.com>. Acesso em: 29 nov. 2003.
4. W3C. **Extensible Markup Language (XML)**. Disponível em: <http://www.w3.org/XML>. Acesso em: 29 nov. 2003.
5. EMARKETER. **PDA Market Report**. Disponível em: <http://www.emarketer.com/products/report.php?pda_market>. Acesso em: 20 jan 2004.
6. ANATEL. **Dados Relevantes do SMC/SMP**. Disponível em: <http://www.anatel.gov.br/Tools/frame.asp?link=/comunicacao_movel/smc/dados_relevantes_smc_smp.pdf>. Acesso em: 1 dez. 2003.
7. ABI RESEARCH. **Traditional PDA vs. Connected PDA Shipments Wolrd Market, Agressive Forecasts: 2002-2008**. Disponível em: <http://www.bargainpda.com/default.asp?newsID=1771&showComments=true>. Acesso em: 20 jan. 2004.
8. MATEUS, Geraldo R.; LOUREIRO, Antônio. **Introdução a Computação Móvel**. 11a. Escola de Computação, Rio de Janeiro, Julho 1998.

9. COMPASS TECHNOLOGIES. **History of Cellular Phones**. Disponível em: <http://www.affordablephones.net/HistoryCellular.htm>. Acesso em: 20 fev. 2004.

10. ANATEL. **A História do Telefone no Brasil: 1990 – Telefonia Celular**. Disponível em: <http://www.anatel.gov.br/biblioteca/Publicacao/museu_telefone/1990.asp>. Acesso em: 5 dez. 2003.

11. DEITEL, Harvey M, et al. **Wireless Internet & Mobile Business: How to Program**. Upper Saddle River, NJ: Prentice Hall, 2002

12. ROCHOL, Juergen, PUFAL, H, BARCELOS, M B. **Análise Comparativa dos Sistemas AMPS, TDMA e CDMA de Telefonia Celular Móvel para Serviços de Comunicação de Dados**. In: XIII SIMPÓSIO BRASILEIRO DE TELECOMUNICAÇÕES, 1995, Águas de Lindoia - SP, p. 557-562. Disponível em: <http://labcom.inf.ufrgs.br/artigos/cdados_redes.pdf>. Acesso em: 26 dez. 2003.

13. MICROSOFT. **Windows Mobile**. Disponível em: <http://www.windowsmobile.com>. Acesso em: 30 jan. 2004.

14. PALMSOURCE. **PalmSource**. Disponível em: <http://www.palmsource.com>. Acesso em: 08 out. 2003.

15. PCTECHGUIDE. **COMPONENTS/MOBILE COMPUTING**. Disponível em: <http://www.pctechguide.com/25mob3.htm>. Acesso em: 10 jan. 2004..

16. APPLE COMPUTER. **Apple Discontinues Development of Newton OS**. Disponível em: <http://www.apple.com/pr/library/1998/feb/27newton.html>. Acesso em: 10 fev. 2004.

17. APPLE COMPUTER. **iPod**. Disponível em: <http://www.apple.com/ipod/>. Acesso em: 30 dez. 2003.

18. PALMONE. **PalmOne Inc**. Disponível em: <http://www.palmone.com>. Acesso em: 30 dez. 2003.

19. MICROSOFT. **Windows CE .NET Documentation**. Disponível em: <http://msdn.microsoft.com/library/default.asp?url=/library/en-us/wcelib40/html/pb_start.asp>. Acesso em: 20 jan. 2004.

20. PALMSOURCE. **Palm OS 5: The Foundation for the Future of Mobile Computing**. Disponível em: <http://www.palmsource.com/includes/palmos5.pdf>. Acesso em: 23 nov. 2003.

21. PALMSOURCE. **Palm OS 5 Programmer´s Companion. Vol 1**. Disponível em: <http://www.palmos.com/dev/support/docs/palmos/CompanionTOC.html>. Acesso em: 20 nov. 2003.

22. ALBERTIN, Alberto Luiz. **Comércio Eletrônico: Modelo, Aspectos e Contribuições de sua Aplicação**. São Paulo: Editora Atlas, 2001

23. DORMAN, Andy. **Wirelles Comunication: O Guia Essencial de Comunicação Sem Fio**. Rio de Janeiro, RJ: Editora Campus, 2001

24. HE, Hao. **What is Service Oriented Architecture?**. Disponível em: <http://webservices.xml.com/pub/a/ws/2003/09/30/soa.html>. Acesso em: 10 fev. 2004.

25. MICROSOFT. **Distributed Component Object Model (DCOM)**. Disponível em: <http://www.microsoft.com/com/tech/dcom.asp>. Acesso em: 29 nov. 2003.

26. OMG. **Common Object Request Broker Architecture (CORBA)**. Disponível em: <http://www.omg.org/gettingstarted/corbafaq.htm>. Acesso em: 29 nov. 2003.

27. SUN MICROSYSTEMS. **Java Remote Method Invocation (RMI)**. Disponível em: <http://java.sun.com/products/jdk/rmi>. Acesso em: 29 nov. 2003.

28. W3C. **UDDI Version 3.0.1**. Disponível em: <http://uddi.org/pubs/uddi_v3.htm>. Acesso em: 29 nov. 2003.

29. KIRK, C. **XML Black Book**. Rio de Janeiro, RJ: Ed. Campus, 1999

30. HAROLD, E. R.. **XML Bible**. Foster City, CA, EUA: IDG Books, 1999

31. IETF. **RFC1057 - Remote Procedure Call (RPC)**. Disponível em: <http://www.ietf.org/rfc/rfc1057.txt>. Acesso em: 29 nov. 2003.

32. W3C. **Simple Object Access Protocol (SOAP) 1.1**. Disponível em: <http://www.w3.org/TR/SOAP>. Acesso em: 29 nov. 2003.

33. IBM, MICROSOFT. **Web Services Security (WS-Security)**. Disponível em: <http://www-106.ibm.com/developerworks/webservices/library/ws-secure/>. Acesso em: 29. nov. 2003.

34. EA, IBM, MICROSOFT, TIBCO SOFTWARE. **Web Services Reliable Messaging Protocol (WS-ReliableMessaging)**. Disponível em: <http://msdn.microsoft.com/ws/2003/03/ws-reliablemessaging>. Acesso em: 29. nov. 2003.

35. IETF. **RFC2616 - Hypertext Transfer Protocol. (HTTP)**. Disponível em: <http://www.ietf.org/rfc/rfc2616.txt>. Acesso em: 29 nov. 2003.

36. IETF. **RFC2774 - An HTTP Extension Framework**. Disponível em: <http://www.ietf.org/rfc/rfc2774.txt>. Acesso em: 29 nov. 2003.

37. IETF. **RFC821 - Simple Mail Transfer Protocol (SMTP)**. Disponível em: <http://www.ietf.org/rfc/rfc0821.txt>. Acesso em: 29 nov. 2003.

38. SUN MICROSYSTEMS. **Java Message Service (JMS)**. Disponível em: <http://java.sun.com/products/jms>. Acesso em: 29 nov. 2003.

39. W3C. **XML Schema**. Disponível em: <http://www.w3.org/XML/Schema>. Acesso em: 30 nov. 2003.

40. WS-I. **Basic Profile Version 1.0a**. Disponível em: <http://www.ws-i.org/Profiles/Basic/2003-08/BasicProfile-1.0a.html>. Acesso em: 30 nov. 2003.

41. W3C. **SOAP Messages with Attachments**. Disponível em: <http://www.w3.org/TR/SOAP-attachments>. Acesso em: 29 nov. 2003.

42. IETF. **RFC1341 - Multipurpose Internet Mail Extensions (MIME)**. Disponível em: <http://www.ietf.org/rfc/rfc1341.txt>. Acesso em: 29 nov. 2003.

43. IETF. **Direct Internet Message Encapsulation (DIME)**. Disponível em: <http://www-106.ibm.com/developerworks/webservices/library/ws-dime>. Acesso em: 29 nov. 2003.

44. IETF. **WS-Attachments**. Disponível em: <http://msdn.microsoft.com/library/en-us/dnglobspec/html/draft-nielsen-dime-soap-01.txt>. Acesso em: 29 nov. 2003.

45. W3C. **Web Services Description Language (WSDL) 1.1**. Disponível em: <http://www.w3.org/TR/wsdl>. Acesso em: 29 nov. 2003.

46. W3C. **XML Schema Part 2: Datatypes**. Disponível em: <http://www.w3.org/TR/xmlschema-2>. Acesso em: 30 nov. 2003.

47. PEREIRA Filho, Nélio Alves. **Linux, Clusters e Alta Disponibilidade**. Disponível em: <http://www.ime.usp.br/~nelio/publications/linuxha/linuxha.pdf.gz>. Acesso em: 26 jun. 2003.

48. MERKEY, Phillip. **Beowulf Project at CESDIS**. Disponível em: <http://cesdis.gsfc.nasa.gov/linux/beowulf/beowulf.html>. Acesso em: 29 jun. 2003.

49. BECKER, Donald. **Beowulf Software**. Disponível em: <http://cesdis.gsfc.nasa.gov/linux/beowulf/software/software.html>. Acesso em: 29 jun. 2003.

50. Ridge, Daniel; Becker, Donald; Merkey, Phillip; Sterling, Thomas. Beowulf: Harnessing the Power of Parallelism in a Pile-of-PCs. **IEEE Aerospace**, 1997.

51. LDP. **The Linux Documentation Project (LDP)**. Disponível em: <http://sunsite.unc.edu/LDP/HOWTO>. Acesso em: 29 jun. 2003.

52. ROBINS, Daniel. **OpenMosix**. Disponível em: <http://cedar.intel.com/cgi-bin/ids.dll/content/content.jsp?cntKey=Generic+Editorial%3a%

3axeon_openmosix&cntType=IDS_EDITORIAL&catCode=BMB>. Acesso em: 29 jun. 2003.

53. BAR, Moshe. **Linux Clusters State of The Art**. Disponível em: <http://www.cineca.it/streaming/openmosix/slides_moshe.php?radice=Diapositiva&last=32>. Acesso em: 29 jun. 2003.

54. MOSIX. **Mosix Inc.**. Disponível em: <http://www.mosix.com>. Acesso em: 29 jun. 2003.

55. OPENMOSIX. **The OpenMosix Project**. Disponível em: <http://openmosix.sourceforge.net>. Acesso em: 29 jun. 2003.

56. OPENSSI. **OpenSSI Project.**. Disponível em: <http://www.openssi.org>. Acesso em: 29 jun. 2003.

57. OSCAR. **Open Source Cluster Application Resources (OSCAR)**. Disponível em: <http://oscar.sf.net>. Acesso em: 26 jun. 2003.

58. IETF. **Network File System (NFS) version 4 Protocol**. Disponível em: <http://www.ietf.org/rfc/rfc3530.txt>. Acesso em: 10 fev. 2004.

59. C3. **Cluster Command & Control (C3) Tool Suite**. Disponível em: <http://www.csm.ornl.gov/torc/C3/Papers/pdcp-v2.0.pdf>. Acesso em: 26 jun. 2003.

60. SAMBA. **Rsync**. Disponível em: <http://www.rsync.samba.org>. Acesso em: 26 jun. 2003.

61. OPENSSL. **OpenSSL**. Disponível em: <http://www.openssl.org>. Acesso em: 26 jun. 2003.

62. OPENSSH. **OpenSSH**. Disponível em: <http://www.openssh.com>. Acesso em: 26 jun. 2003.

63. ISC. **Dynamic Host Configuration Protocol (DHCP)**. Disponível em: <http://www.isc.org/products/DHCP>. Acesso em: 26 jun. 2003.

64. BALD GUY SOFTWARE. **Systemimager**. Disponível em: <http://www.systemimager.org>. Acesso em: 26. jun. 2003.

65. LAM. **LAM / MPI Parallel Computing**. Disponível em: <http://www.lam-mpi.org>. Acesso em: 26 jun. 2003.

66. LUI. **Linux Utility for Cluster Installation (LUI)**. Disponível em: <http://oss.software.ibm.com/developerworks/projects/lui>. Acesso em: 26 jun. 2003.

67. MAUI. **Maui PBS Scheduler**. Disponível em: <http://supercluster.org/maui>. Acesso em: 26. jun. 2003.

68. MPICH. **A High-Performance, Portable Implementation of the MPI Message Passing Interface Standard**. Disponível em: <http://www-unix.mcs.anl.gov/mpi/mpich/papers/mpicharticle/paper.html>. Acesso em: 27 jun. 2003.

69. IETF. **RFC318 - Telnet Protocol**. Disponível em: <http://www.ietf.org/rfc/rfc318.txt>. Acesso em: 28 jun. 2003.

70. IETF. **RFC1282 - BSD Rlogin**. Disponível em: <http://www.ietf.org/rfc/rfc1282.txt>. Acesso em: 28. jun. 2003.

71. IETF. **RFC959 - File Transfer Protocol (FTP)**. Disponível em: <http://www.ietf.org/rfc/rfc959.txt>. Acesso em: 28. jun. 2003.

72. OPENPBS. **OpenPBS**. Disponível em: <http://www.openpbs.org>. Acesso em: 28 jun. 2003.

73. PBS. **The Portable Batch System (PBS)**. Disponível em: <http://www.inf.ufrgs.br/labtec/pbs.html>. Acesso em: 28 jun. 2003.

74. PVM. **Parallel Virtual Machine (PVM)**. Disponível em: <http://www.csm.ornl.gov/pvm>. Acesso em: 28 jun. 2003.

75. SIS. **SIS Paper**. Disponível em: <http://www.sisuite.org/presentations/OLS_2002/sispaper.pdf>. Acesso em: 28 jun. 2003.

76. Bryhni, Haakon. A comparison of load balancing techiniques for scalable Web Servers. **IEEE Network**, Agosto 2000.

77. EXTREME NETWORKS. **Extreme Networks TechBrief - Server Load Balancing**. Disponível em: <http://www.extremenetworks.com/libraries/whitepapers/technology/Server_load_balancing.pdf>. Acesso em: 05 dez. 2003.

78. IETF. **RFC1035 – Domain Names – Implementation and Specification**. Disponível em: <http://www.ietf.org/rfc/rfc1035.txt>. Acesso em: 14 dez. 2003.

79. NETSCAPE. **The SSL Protocol Version 3.0**. Disponível em: <http://wp.netscape.com/eng/ssl3/draft302.txt>. Acesso em: 30 nov. 2003.

80. KUNAL, Dua. **Balance your Web Server's Load**. Disponível em: <http://pcquest.ciol.com/content/technology/101071402.asp>. Acesso em: 30 nov. 2003.

81. IETF. **DNS Support for Load Balancing**. Disponível em: <http://www.ietf.org/rfc/rfc1794.txt>. Acesso em: 30 nov. 2003.

82. Colajanni, Michele. Analysis of Task Assignment Policies in Scalable Distributed Web-server Systems. **IEEE Transactions on Parallel and Distributed Systems**, Junho 1998.

83. Schemers, R.J. lbmnamed: A load balancing name server in Perl. **9th System Administration Conference (LISA)**, 1995.

84. PalmSource. **Palm OS 5: The Foundation for the Future of Mobile Computing**. Disponível em: <http://www.palmsource.com/includes/palmos5.pdf>. Acesso em: 23 nov. 2003.

85. PHP GROUP. **PHP: Hypertext Preprocessor**. Disponível em: <http://www.php.net>. Acesso em: 15 dez. 2003.

86. PEAR GROUP. **PHP Extension and Application Repository (PEAR)**. Disponível em: <http://pear.php.net>. Acesso em: 29 nov. 2003.

87. APACHE SOFTWARE FOUNDATION. **Apache HTTP Server Project** . Disponível em: <http://httpd.apache.org/>. Acesso em: 12 dez. 2003.

88. MySQL AB. **MySQL**. Disponível em: <http://www.mysql.com>. Acesso em: 15 dez. 2003.

89. FREEBSD PROJECT. **FreeBSD Project**. Disponível em: <http://www.freebsd.org>. Acesso em: 12 dez. 2003.

90. LARMAN, Craig. **Utilizando UML e Padrões**. Porto Alegre: Bookman, 2001

91. PCR-TOOLS PROJECT. **Palm OS programming with GCC**. Disponível em: <http://prc-tools.sourceforge.net/doc/>. Acesso em: 10 fev. 2004.