**VÂNIA DE CARVALHO MARÇAL**

# PROPOSTA DE UMA APLICAÇÃO USANDO BANCO DE DADOS DISTRIBUÍDOS ORACLE9*i* E A TECNOLOGIA WAP DAS REDES *WIRELESS*

Monografia de conclusão do curso de Ciência da Computação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências da disciplina Projeto Orientado II, para obtenção do título de Bacharel em Ciência da Computação.

Orientadora
Profª. Olinda Nogueira Paes Cardoso

Co-Orientador
Prof. Ricardo Martins de Abreu Silva

LAVRAS
MINAS GERAIS – BRASIL
2002

**VÂNIA DE CARVALHO MARÇAL**

# PROPOSTA DE UMA APLICAÇÃO USANDO BANCO DE DADOS DISTRIBUÍDOS ORACLE9*i* E A TECNOLOGIA WAP DAS REDES *WIRELESS*

Monografia de conclusão do curso de Ciência da Computação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências da disciplina Projeto Orientado II, para obtenção do título de Bacharel em Ciência da Computação.

APROVADA, em 16 de dezembro de 2002.

_____
Profª Olinda Nogueira Paes Cardoso
(Orientadora)

_____
Prof. Ricardo Martins de Abreu Silva
(Co-Orientador)

LAVRAS
MINAS GERAIS – BRASIL
2002

*"O maior heroísmo de um jovem é manter o sorriso no rosto, mesmo que o coração estraçalhe de dor."*

(Autor Desconhecido)

Dedico este trabalho a todos que, de alguma forma, deram-me motivos para que eu mantivesse o sorriso no rosto mesmo quando meu coração se estraçalhava de dor. E, em especial, dedico à minha maravilhosa mãe, que partiu antes que este momento chegasse. Hoje, especialmente, a saudade é bem mais forte, mas a lembrança de sua voz amiga, de seu sorriso, de seu abraço forte, realimentam o amor que tornou- me capaz de chegar até aqui e de concretizar este nosso sonho.

# AGRADECIMENTOS

Agradeço a Deus pela presença fundamental, pela força e principalmente, por amparar-me em todos os momentos difíceis. Expresso também meu maior agradecimento e o meu profundo respeito, os quais sempre serão poucos diante do muito que foi oferecido, àquela que quando deveria ser simplesmente professora e orientadora, mostrou-se um exemplo de dedicação, doação, força, compreensão e, sobretudo de amor. Obrigada, Olinda! Esteja certa de que aonde quer que eu vá, levarei você para sempre no meu coração.

## RESUMO

O objetivo deste trabalho é conciliar as tecnologias de Banco de dados Distribuído e WAP (protocolo para redes sem fios), usando como exemplo de aplicação de um Sistema de Pronto atendimento Remoto Hospitalar. Este sistema deve atender às necessidades estratégicas, econômicas e sociais de determinada região, evidenciando como integrar e acessar informações distribuídas em diversas localidades, através de dispositivos sem fio, tais como telefones celulares. A integração dos sistemas hospitalares via Internet e o acesso a eles, via rede Wireless (sem fio) é apresentada dentro do contexto do Sistema Gerenciador de Bancos de Dados Oracle9*i*, considerando-se a funcionalidade da arquitetura cliente-servidor. Os resultados foram obtidos de um estudo sobre as técnicas de implementação deste sistema, e as dificuldades em seu desenvolvimento foram apresentadas.



**Palavras-chave:** Banco de Dados Distribuídos, Redes *Wireless*, Dispositivos Sem Fio, Telefone Celular, Protocolo WAP, Oracle9*i*, Sistema Hospitalar.

**SUMÁRIO**

## LISTA DE ABREVIATURAS

BDD – Banco de Dados Distribuídos

CDMA – *Code Division Multiple Access*

CDM – *Common Data Model*

CGI – *Commom Gateway Interface*

CPU – *Central Processing Unit*

DBA – *Data Base Administrator*

E / S – Entrada e Saída

GSM – *Global System for Mobile Communication*

HTML – *HyperText Markup Language*

HTTP – *Hypertext Markup Protoco*l

IP – *Internet Protocol*

ISO – *International Standards Organization*

LAN – *Local Area Network*

PAP – *Push Access Protocol*

PC – *Personal Computer*

PDA – *Personal Digital Assistant*

PHP – *Personal Home Page*

PPG – *Push Proxy Gateway*

OTA – *Over The Air*

SAI - (*Session Initiation Application)*

SBDF – Sistema de Banco de Dados Federados

SGBD – Sistema Gerenciador de Banco de Dados

SGBDD – Sistema Gerenciador de Banco de Dados Distribuídos

SI – *Service Indication*

SL – *Service Loading*

SMS – *Short Message Service*

SQL – *Structured Query Language*

SSL – *Secure Sockets Layer*

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

TDMA – *Time Division Multiple Access*

TLS – *Transport Layer Security*

UDP – *User Datagram Protocol*

UI – *User Interface*

URI – *Universal Resource Identifier*

URL – *Universal Resource Locator*

XHTML – *eXtensible HyperText Markup Language*

XML – *eXtensible markup Language*

WAN – *Wide Area Network*

WAE – *Wireless Application Environment*

WAP – *Wireless Application Protocol*

WCMP – *Wireless Control Message Protocol*

WDP – *Wireless Datagram Protocol*

WIM – *WAP Identity Module*

WML – *Wireless Markup Language*

WSP – *Wireless Session Protocol*

WTA – *Wireless Telephony Application*

WTAI – *Wireless Telephony Application Interface*

WTLS – *Wireless Transport Layer Security*

WTP – *Wireless Transaction Protocol*

WWW – *World Wide Web*

2,5G – Segunda geração de celulares

3G – Terceira geração de celulares

## LISTA DE FIGURAS

## LISTA DE TABELAS

# CAPÍTULO 1 INTRODUÇÃO

Nas últimas décadas, a Tecnologia da Informação evoluiu dos primeiros computadores centrais até os atuais sistemas distribuídos. Essa visão moderna e descentralizada busca obter vantagens, principalmente em termos de acessibilidade, disponibilidade e custo. Um importante componente desses sistemas distribuídos consiste no Banco de Dados Distribuídos (BDD).

Nos últimos anos, houve diversas tentativas de viabilizar o acesso à Internet em uma plataforma sem fio. Neste contexto, o WAP (*Wireless Application Protocol* - Protocolo para Aplicações Sem Fio) surgiu como promessa de um protocolo capaz de reconhecer os serviços WWW (*World Wide Web*), além de propor novos, por exemplo o serviço de mensagens telefônicas. Esse padrão especifica um ambiente de aplicação e uma pilha de protocolos para esses dispositivos sem fio como, por exemplo, telefones celulares. Isto permite às aplicações tirarem proveito do fato de que, freqüentemente, o usuário de Internet possui um dispositivo de telefonia móvel [Spo02].

O objetivo deste trabalho é conciliar estas duas tecnologias, Banco de Dados Distribuídos e a tecnologia WAP, numa aplicação que atenda às necessidades estratégicas, econômicas e sociais de determinada região. Dessa forma, seria possível integrar informações distribuídas em diversas localidades de uma região e acessá-las em qualquer parte através dos dispositivos sem fio.

Portanto, a intenção deste projeto é integrar sistemas autônomos e heterogêneos via *Internet e* acessá-los via rede *Wireless.*

A aplicação hospitalar refere-se à viabilização de dispositivos sem fio, por exemplo, telefones celulares em ambulâncias, que estejam conectados através da Internet a hospitais da região. O sistema irá buscar o hospital que melhor satisfaça às necessidades de determinado caso que esteja sendo atendido. Esta estratégia permite que sejam feitas consultas como, por exemplo, qual hospital está apto a receber dado paciente, em dada situação. Além disso, a reserva de um leito para um dos hospitais poderá ser realizada, a fim de que, ao chegar ao hospital preferencial, já exista uma equipe médica o aguardando para o devido atendimento.

A possível modelagem do Banco de Dados Distribuído, que irá gerenciar os bancos de dados locais de cada hospital, interagindo via WAP com o telefone celular localizado na ambulância, poderia ser como a apresentada na Figura 1.

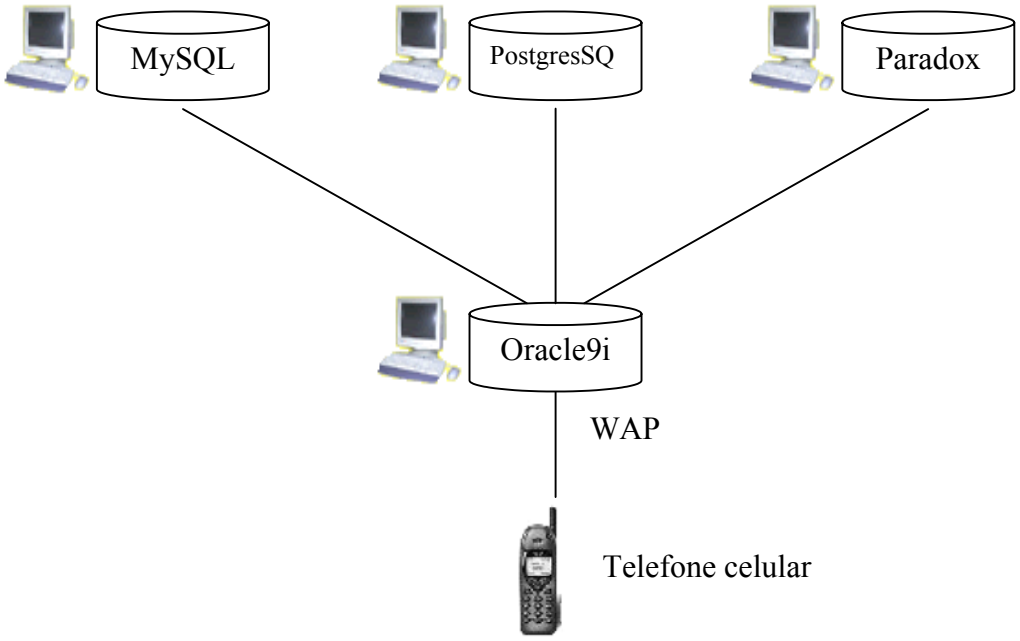



**Figura 1**: SGBDD's Heterogêneos conectados ao telefone celular via WAP

Inserido na conjuntura atual, em que o telefone celular tornou-se um dispositivo quase indispensável e, acima de tudo, acessível à maioria das pessoas, é amplamente viável investir neste tipo serviço. Pois além de gerar uma economia que estará sendo feita ao evitar deslocamentos desnecessários, exclui o risco do paciente chegar a um local onde não haja o devido atendimento.

O segundo capítulo, apresenta conceitos da arquitetura cliente-servidor e relaciona os mesmos a bancos de dados distribuídos, descrevendo algumas das

facilidades no Oracle9*i* para suportar banco de dados distribuídos. No terceiro capítulo, são tratados alguns aspectos gerais sobre o WAP e descreve-se sobre o funcionamento desse protocolo para redes sem fio, sob um ponto de vista teórico, isto é, com abordagem puramente centrada em suas camadas e nos serviços por estas oferecidos. No quarto capítulo, as duas tecnologias são relacionadas, mostrando o Oracle9*i* como solução *Wireless.* No quinto capítulo, são apresentados os procedimentos tomados para a construção do Banco de Dados da aplicação, como poderia ser feita a integração dos hospitais no servidor *Web*, e as ferramentas utilizadas para desenvolvimento e teste da aplicação. E, finalmente, o sexto capítulo, apresenta a conclusão deste trabalho, referenciando não só os pontos que exigiram maior esforço e disponibilidade de tempo, bem como aqueles que ofereceram menor grau de dificuldade de desenvolvimento e propostas de trabalhos futuros.

## CAPÍTULO 2 BANCO DE DADOS DISTRIBUÍDOS

Da mesma maneira, a Oracle não é só uma plataforma de desenvolvimento de aplicações, ela fornece toda uma infra-estrutura técnica de desenvolvimento e gerenciamento de aplicações móveis. Logo, todo o referencial teórico sobre como desenvolver uma aplicação móvel usando a versão Oracle9*i* foi retirado do *site* oficial da Oracle. Indiscutivelmente, ela abrange uma infra-estrutura completa e ideal para empreendimentos que buscam uma maior eficiência com a tecnologia móvel, sendo capaz de distribuir todas as aplicações e dados em qualquer lugar, a qualquer hora, e em qualquer dispositivo.

Os bancos de dados distribuídos mostram as vantagens da computação distribuída sob domínio da gerência de bancos de dados. Um sistema de computação distribuído consiste em uma série de elementos de processamento, não necessariamente homogêneos, que são interligados por um sistema de rede de computadores e que cooperam na realização de determinadas tarefas específicas. Vale ressaltar que a maioria das informações deste capítulo foi retirada de [EN02].

Com objetivo geral, sistemas de computação distribuídos repartem um grande e não gerenciável problema em partes menores e resolvem o mesmo de maneira eficiente e coordenada. A viabilidade econômica dessa abordagem destaca-se por duas razões: (1) maior esforço computacional é despendido para solucionar uma tarefa complexa e (2) cada elemento autônomo do

processamento pode ser gerenciado independentemente e desenvolver suas próprias aplicações.

Define-se Banco de Dados Distribuído (BDD) como uma coleção de vários bancos de dados logicamente inter-relacionados, distribuídos ao longo de um sistema de rede de computadores. Logo, Sistema de Gerência de Banco de Dados Distribuídos (SGBDD) é um conjunto de programas que gerencia um banco de dados distribuído e ao mesmo tempo torna a distribuição transparente para o usuário[Cer85].

Uma coleção de arquivos armazenados em diferentes nós de um sistema de rede e manutenção de inter-relações entre eles através de *hyperlinks* tornaram-se uma organização comum na Internet com os arquivos das páginas da *Web*. As funções comuns no gerenciamento de bancos de dados, incluindo o processamento uniforme de consultas e o processamento uniforme de transações, ainda não se aplicam a esse cenário. A tecnologia, no entanto, está caminhando em uma direção tal que os bancos de dados distribuídos da *World Wide Web* (WWW) se tornarão realidade em um futuro próximo.

## 2.1 Vantagens dos Bancos de Dados Distribuídos

A gerência de bancos de dados distribuídos foi proposta por vários motivos, abrangendo desde a descentralização organizacional e o processamento

econômico até uma maior autonomia. A seguir encontram-se algumas destas vantagens.

### 2.1.1 Gerência de dados distribuídos com diferentes tipos de transparência

Em termos ideais, um SGBD deve ser transparente na distribuição, de certo modo escondendo os detalhes sobre onde cada arquivo (tabela ou relação) está fisicamente armazenado dentro do sistema. Os seguintes tipos de transparências são possíveis:

Transparência de distribuição ou de rede

Refere-se a liberar o usuário dos detalhes sobre a rede. Pode ser dividida entre transparência de localização e transparência de nomeação. A transparência de localização se refere ao fato de que o comando utilizado para realizar uma tarefa é independente da localização dos dados e da localização do sistema no qual o comando foi emitido. A transparência de nomeação implica que, uma vez que se especifique um nome, pode-se acessar os objetos nomeados de maneira não-ambígua sem especificação adicional.

Transparência de replicação

Cópias de dados podem ser armazenadas em vários *sites* para melhor disponibilidade, desempenho e confiabilidade. A transparência de replicação faz

com que o usuário não se torne ciente da existência de cópias.

Transparência de fragmentação

Dois tipos de fragmentação são possíveis: A fragmentação horizontal que distribui uma relação entre conjuntos de tuplas (linhas); e a fragmentação vertical que distribui uma relação ente sub-relações em que cada sub-relação é definida através de um conjunto de colunas da relação original. Uma consulta global feita pelo usuário deve ser transformada em diversos fragmentos de consultas. A transparência de fragmentação faz com que o usuário não se torne ciente da existência de fragmentos.

### 2.1.2 Confiabilidade e disponibilidade crescentes

São duas entre as vantagens potenciais mais comuns citadas para bancos de dados distribuídos. A confiabilidade é geralmente definida como a probabilidade que um sistema esteja funcionando (não esteja parado) em um certo momento, enquanto disponibilidade é a probabilidade de que um sistema esteja continuamente disponível durante um intervalo de tempo. Quando os dados e o SGBD estão distribuídos em diversos *sites*, um *site* pode falhar enquanto outros continuam a operar. Somente os dados e os programas que existem no *site* que falhou não podem ser acessados. Isso melhora tanto a confiabilidade quanto a disponibilidade.

Outros aperfeiçoamentos são conseguidos através de criteriosa replicação de dados e de *software* em mais de um *site*. Em um sistema centralizado falhas em um único *site* fazem com que todo o sistema se torne indisponível para todos os usuários. Em um banco de dados distribuídos, alguns dados podem ser inatingíveis, mas os usuários podem acessar outras partes do banco de dados.

### 2.1.3 Melhor desempenho

Um SGBD distribuído fragmenta o banco de dados mantendo os dados mais próximos do local onde são mais necessitados. A localização de dados reduz a disputa entre serviços da CPU (*Central Processing Unit*) e de E/S (Entrada e Saída) e ao mesmo tempo reduz a demora no acesso que sistemas de rede de áreas distantes (longo alcance) implicam. Quando um grande banco de dados é distribuído em vários *sites*, existem bancos de dados menores em cada *site*. Por esse motivo, consultas e transações locais que acessam dados em um único *site* tem menor número de transações em execução do que no caso de todas as transações serem submetidas a um único banco de dados centralizado. Além disso, o paralelismo inter-consultas e intra-consultas pode ser obtido executando múltiplas consultas em diferentes *sites* ou desmembrando-se uma consulta em uma série de sub-consultas que sejam executadas em paralelo. Isso contribui para um melhor desempenho.

### 2.1.4 Expansão mais fácil

Em um ambiente distribuído, a expansão do sistema em termos de acrescentar mais dados, aumentar o tamanho dos bancos de dados ou acrescentar mais processadores é muito mais fácil.

A transparência total fornece ao usuário global uma visão de todo o Sistema de Banco de Dados Distribuído (SBDD) como se fosse um único sistema centralizado. A transparência é fornecida como um complemento para a autonomia, que fornece aos usuários um controle mais rígido sobre seus próprios bancos de dados locais. Características da transparência podem ser implementadas como uma parte da linguagem do usuário, que pode traduzir os serviços necessários para operações apropriadas. Além disso, a transparência causa impacto sobre as características que devem ser fornecidas pelo sistema operacional e pelo SGBD.

## 2.2 Funções Adicionais de Banco de Dados Distribuídos

A distribuição acarreta maior complexidade no projeto e na implementação do sistema. Para obter as vantagens potenciais listadas anteriormente, o *software* do SGBDD deve ter capacidade de fornecer as seguintes funções, além das funções de um SGBD centralizado:

Controlar os dados

Capacidade de controlar a distribuição, a fragmentação e a replicação de dados expandindo o catálogo do SGBDD.

Processamento de consultas distribuídas

Capacidade de acessar *sites* remotos e transmitir consultas e dados entre os vários *sites* através de uma rede de comunicação.

Gerenciamento de transações distribuídas

Capacidade de planejar estratégias de execução para consultas e transações que acessam dados de mais de um *site* e de sincronizar o acesso a dados distribuídos e manter integridade do banco de dados geral.

Gerenciamento de dados replicados

Capacidade de decidir qual cópia de um item de dado replicado acessar e manter a consistência de cópias de um item de dado replicado.

Recuperação de bancos de dados distribuídos

Capacidade de se recuperar de colapsos (*crash*) em *sites* individuais e de novos tipos de falhas tais como a falha de um *link* de comunicação.

Segurança

Transações distribuídas devem ser executadas com o gerenciamento apropriado da segurança dos dados e os privilégios de autorização / acesso de usuários.

Gerenciamento do diretório (catálogo[1]) distribuído

Um diretório contém informações (metadados) sobre dados do banco de dados. O diretório pode ser geral para todo o BDD ou local para cada *site*. O posicionamento e a distribuição do diretório são questões inerentes a projetos e políticas.

Por si sós, essas funções aumentam a complexidade de um SGBDD em relação a um SGBD centralizado. No nível físico do *hardware*, os seguintes fatores principais distinguem um SGBDD de um sistema centralizado:

Existem muitos computadores, chamados *sites* ou nós. Esses *sites* devem ser conectados por um tipo de rede comunicação para transmitir dados e comandos entre *sites*, conforme mostra a Figura 2

[1] Um catálogo do sistema tem a função de armazenar os esquemas, ou descrições dos bancos de dados que o SGBD mantém. Essas informações geralmente são chamadas de metadados. Elas incluem uma descrição do esquema conceitual do banco de dados, o

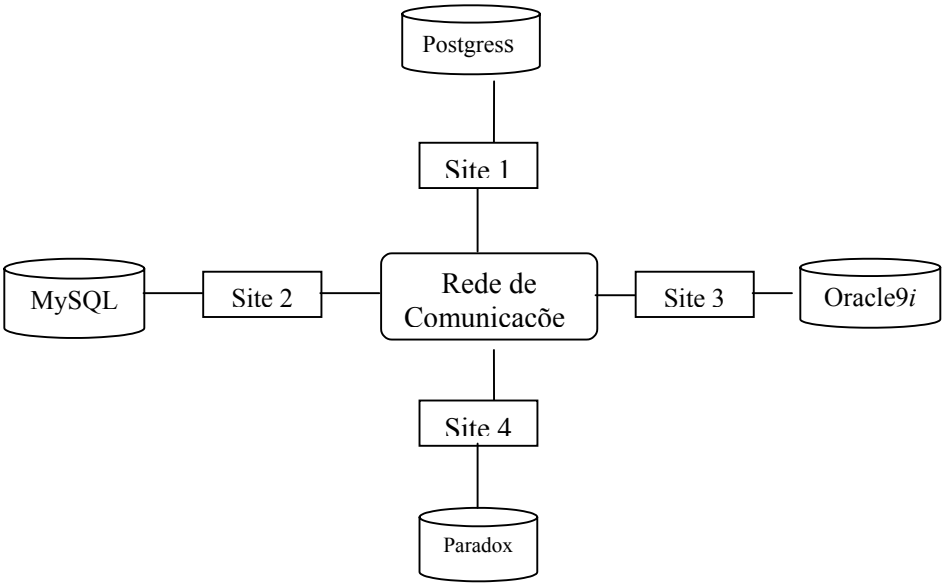



**Figura2**: Arquitetura de um Banco de Dados verdadeiramente distribuído

## 2.3 Fragmentação, Replicação e Alocação de Dados

Em um BDD, devem ser tomadas decisões em relação a qual *site* deve utilizado para armazenar quais partes do banco de dados. Por ora, vamos admitir que não haja nenhuma replicação; ou seja, cada relação - ou parte de relação – deve ser armazenada somente em um *site*. Também utilizaremos a terminologia de banco de dados relacionais – conceitos similares aplicam-se a outros modelos de dados.

Parte-se do pressuposto que estamos iniciando com um esquema de um banco de dados relacional e é preciso decidir como distribuir as relações ao

---

esquema interno, quaisquer esquemas externos, bem como os módulos específicos do

longo dos vários *sites*. Antes de decidir como distribuir os dados, deve-se determinar as unidades lógicas do banco de dados que devem ser distribuídas. As unidades lógicas mais simples são as próprias relações; ou seja, cada relação inteira deve ser armazenada em um determinado *site*. Em muitos casos, no entanto, uma relação pode ser dividida em unidades lógicas menores para fins de distribuição.

A replicação é útil para melhorar a disponibilidade de dados. O caso mais extremo é a replicação de todo o banco de dados em todos os *sites* no sistema distribuído, criando assim uma banco de dados totalmente replicado. Isso pode melhorar bastante a disponibilidade, porque o sistema pode continuar a operar enquanto pelo menos um *site* estiver ligado. Também melhora o desempenho de recuperação para consultas globais, porque o resultado dessa consulta pode ser obtido localmente a partir de qualquer *site* individual; assim sendo, uma consulta de recuperação pode ser processada no *site* local no qual é submetida, se esse *site* incluir um módulo de servidor.

A desvantagem de replicação total é que pode desacelerar drasticamente operações de atualização, uma vez que uma única atualização lógica deve ser realizada em todas as cópias do banco de dados para manter consistentes as cópias do banco de dados.

## 2.4 Tipos de Sistemas de Banco de Dados Distribuídos

---

SGBD.

A expressão sistemas de gerência de bancos de dados distribuídos pode descrever vários sistemas que se distinguem um do outro em muitos aspectos. A principal característica em comum entre estes sistemas é o fato de que os dados e os *softwares* são distribuídos em diversos *sites* conectados por alguma forma de sistema de rede de comunicação[CD91].

É importante considerar o grau de homogeneidade dos *softwares* do SGBDD. Se todos os servidores (ou SGBD locais individuais) utilizam *softwares* idênticos e todos os usuários (clientes) utilizam *softwares* idênticos, o SGBDD é chamado homogêneo; em caso contrário é chamado heterogêneo. Um outro fator relacionado ao grau de homogeneidade é o grau de autonomia local. Se não existir previsão para que o *site* local funcione como um SGBD autônomo (*stand-alone*), o sistema não tem autonomia local.

Por outro lado, se o acesso direto de transações locais a um servidor for permitido, o sistema tem algum grau de autonomia local. Em um extremo do espectro de autonomia, temos um SGBDD que "se parece" com um SGBD centralizado para o usuário. Existe um único esquema conceitual e todo acesso ao sistema é obtido através de um *site* que é parte de um SGBDD chamado SGBDD federado (ou um sistema de múltiplos bancos de dados). Nesse sistema, cada servidor é um SGBD centralizado independente e autônomo que tem seus próprios usuários locais, transações locais e DBA (*Data Base Administrator*) e, portanto tem um grau muito elevado de autonomia local.

A expressão sistema de Banco de Dados Federado (SBDF) é utilizada quando existe alguma visão ou esquema global da federação de bancos de dados que sejam compartilhados pelas aplicações. Por outro lado, um sistema de múltiplos bancos de dados não tem um esquema global e constrói esse esquema interativamente, à medida que se fizer necessário por parte da aplicação.

Ambos os sistemas são híbridos entre sistema distribuídos e centralizados e a distinção que faz-se entre eles não é estritamente seguida. Em um SBDD heterogêneo, um servidor pode ser um SGBD relacional, um outro pode ser um SGBD de rede e um terceiro pode ser um SGBD hierárquico. Os diferentes tipos de autonomia contribuem para uma heterogeneidade semântica, a qual deve ser resolvida em um SBDF heterogêneo.

A heterogeneidade semântica ocorre quando existem diferenças no significado, na interpretação e na utilização pretendida dos mesmos dados ou de dados correlacionados. A heterogeneidade semântica entre sistemas de bancos de dados componentes (SBD) cria o maior obstáculo quando se projetam esquemas globais de bancos de dados heterogêneos.

A autonomia de projeto de SBD componentes refere-se à sua liberdade de escolher os seguintes parâmetros de projeto, que por sua vez afetam a eventual complexidade do SBDF:

1) O universo de discurso do qual os dados são extraídos podem conter algumas informações comuns e algumas inteiramente distintas.

2) A representação e nomeação de elementos de dados e a estrutura do modelo de dados podem ser previamente especificadas para cada banco de dados local.

3) Entendimento, significado e interpretação subjetiva dos dados constituem-se nos mais fortes contribuintes para a heterogeneidade semântica.

4) Restrições de transações e outras políticas de transações.

5) Resumos, agregações e outras características e operações de processamentos de dados mantidas pelo sistema.

A autonomia de comunicação de um SBD componente refere-se à sua capacidade de decidir comunicar-se com um outro SBD componente. A autonomia de execução refere-se à capacidade de um SBD componente executar operações locais sem interferência de operações externas de outros SBD componentes, e à sua capacidade de decidir sobre a ordem na qual executá-las.

A autonomia de associação de um SBD componente implica que ele tem a capacidade de decidir se deve e quanto deve compartilhar sua funcionalidade (operações que suporta) e recurso (dados que gerencia) com outros SBD componentes.

O maior desafio quando se projetam SBDF é deixar que SBD componentes interoperem, ao mesmo tempo, fornecendo a eles os tipos mencionados de autonomia. O esquema local é o esquema conceitual (definição completa do banco de dados) de um banco de dados componente, e o esquema componente é extraído traduzindo-se o esquema local par um modelo de dados

canônico ou modelo de dados comum (CDM - *Common Data Model*) para o SBDF. O translado de esquemas desde o esquema local até o esquema componente é acompanhado pela geração de mapeamentos para transformar comandos em um esquema componente em comandos no esquema local correspondente. O esquema de exportação representa o subconjunto de um esquema componente que esteja disponível para o SBDF. O esquema federado é o esquema ou visão global, que resulta da integração de todos os esquemas de exportação compartilháveis. Os esquemas externos definem o esquema para um grupo de usuários ou para uma aplicação como na arquitetura de três níveis.

## 2.5 Visão Geral de Controle de Concorrência e recuperação em BDD

O método de controle de concorrência é responsável por manter consistência entre as várias cópias dos itens de dados. Já o método de recuperação é responsável por tornar uma cópia consistente com outras cópias se o *site* no qual a cópia estiver armazenada falhar e se recuperar posteriormente.

O sistema deve ter capacidade de lidar com falhas em um ou mais dos *links* de comunicação que conectam os *sites*. Um caso extremo desse problema é que pode ocorrer um particionamento da rede.

### 2.5.1 Controle de concorrência distribuída baseada em cópia distinta de um item de dado

Para lidar com itens de dados replicados em um banco de dados distribuído, têm sido propostos muitos métodos de controle concorrência que estendem as técnicas de controle de concorrência para bancos de dados centralizados. A idéia é designar uma determinada cópia de cada item de dado como uma cópia distinta. Os bloqueios para esse item de dado são associados à cópia distinta e todas as solicitações de bloqueio (*locking*) e desbloqueio (*unlocking*) são enviadas ao *site* que contém essa cópia.

### 2.5.2 Controle de concorrência distribuída baseada em votação

Neste método uma solicitação de bloqueio é enviada a todos os *sites* que incluem uma cópia do item de dado. Cada cópia mantém seu próprio bloqueio e pode conceder ou negar o pedido. O método de votação é considerado um método de controle de concorrência verdadeiramente distribuída, uma vez que a responsabilidade por uma decisão está concentrada em todos os *sites* envolvidos.

### 2.5.3 Recuperação Distribuída

O processo de recuperação em bancos de dados distribuídos é bastante complexo. Em alguns casos, é bastante difícil até mesmo determinar se um *site* está desativado (fora do ar), sem trocar inúmeras mensagens com outros *sites*.

Outro problema com relação à recuperação distribuída é o *commit*[2] distribuído. Quando uma transação está atualizando dados em diversos *sites*, ela não pode dar *commit* até ter certeza de que o efeito da transação em todos os *sites* não pode ser perdido. Geralmente, o protocolo *commit* de duas fases (*two-phase commit*) é utilizado para garantir a precisão do *commit* distribuído.

## 2.6 Visão Geral da Arquitetura Cliente-Servidor e sua relação com Banco de Dados Distribuídos

Aplicações de bancos de dados distribuídos estão sendo desenvolvidas no contexto da arquitetura cliente-servidor. O modo exato de dividir a funcionalidade do SGBD entre cliente e servidor ainda não foi estabelecido. Diferentes abordagens têm sido propostas. Uma possibilidade é incluir a funcionalidade de um SGBD centralizado no nível do servidor.

Uma série de produtos de SGBD relacional tem adotado essa abordagem na qual um servidor SQL é fornecido aos clientes. Cada cliente deve então formular as consultas apropriadas da SQL e fornecer as funções de interface do usuário e de interface da linguagem de programação.

Uma vez que a SQL é um padrão relacional, vários servidores SQL, possivelmente fornecidos por vendedores distintos, podem aceitar comandos da

[2] Um comando *commit* efetiva todas as alterações feitas na base de dados desde o último *commit* ou *roolback* ou desde a conexão inicial do usuário, se não houveram

SQL. O cliente pode também consultar um dicionário de dados que inclui informações sobre a distribuição de dados entre os vários servidores SQL, bem como módulos para decomposição de uma consulta global em uma série de consultas locais que podem ser executados nos vários *sites*.

A interação entre cliente e servidor pode proceder da seguinte maneira durante o processamento de uma consulta na SQL :

**1.** O cliente analisa uma consulta do usuário e a decompõe em uma série de consultas em *sites* independentes. Cada consulta do *site* é enviada para o *site* do servidor apropriado.

**2.** Cada servidor processa a consulta local e envia a relação resultante para o *site* do cliente.

**3.** O *site* cliente combina os resultados das subconsultas para produzir o resultado da consulta originalmente submetida.

A interação entre o cliente e o servidor pode ser especificada pelo usuário no nível do cliente ou através de um módulo cliente especializado do SGBD. Em um SGBDD típico, é comum dividir os módulos de *software* em níveis:

**1.** O *software* do servidor é responsável pelo gerenciamento de dados locais em um *site*, de maneira muito parecida com softwares de SGBD centralizados.

---

comandos realizados. O comando *commit* é aplicável a todos os comandos SQL, incluindo comandos de definição e controle.

**2.** O software do cliente é responsável pela maior parte das funções de distribuição; acessa informações sobre a distribuição de dados a partir do catálogo do SGBDD e processa todas as solicitações que requerem acesso a mais de um *site*. Também lida com todas as interfaces de usuários.

**3.** O *software* de comunicação (às vezes em conjunto com um sistema operacional distribuído) fornece os meios de comunicação que são utilizados pelo cliente para transmitir comandos e dados entre os vários *sites*, conforme necessário. Isso na é estritamente parte do SGBDD, mas fornece os meios e serviços de comunicação essenciais.

Alguns SGBDD não oferecem transparência de distribuição; em vez disso, exigem que usuários estejam atentos aos detalhes sobre a distribuição de dados.

## 2.7 Banco de Dados Distribuídos no Oracle

Na arquitetura cliente-servidor, o sistema de banco de dados Oracle é dividido em duas partes: (1) um *front-end* como a parte do cliente e (2) um *back-end* como a parte do servidor.

A parte do cliente é uma aplicação do banco de dados *front-end* que interage com o usuário. O cliente tem a responsabilidade de acesso aos dos e meramente cuida da solicitação, do processamento e da apresentação de dados gerenciados pelo servidor.

O servidor executa o Oracle e cuida das funções relacionadas com o acesso compartilhado concorrente. Aceita instruções da SQL e PL/SQL[3] originárias de aplicações de clientes, processa as mesmas e envia os resultado de volta para o cliente.

Aplicações cliente-servidor do Oracle oferecem transparência de localização, tornando a localização de dados transparente para usuários; várias características como visões, sinônimos e procedimentos contribuem para isso. A nomeação global é obtida utilizando-se <nome_da_tabela@nome_do_banco_de_dados> para fazer referência unicamente a tabelas.

Todos os bancos de dados do Oracle em um sistema de banco de dados distribuído (SGBDD) utilizam o *software* de rede Net8 para comunicação inter-banco de dados. O Net8 permite que bancos de dados se comuniquem através de redes para suportar transações remotas e distribuídas. Consolida instruções da SQL em um dos muitos protocolos de comunicação para facilitar comunicações de cliente para servidor e em seguida consolida os resultados de volta da mesma maneira, para o cliente [CD91].

[3] PL/SQL é a extensão da linguagem procedural do Oracle para SQL. Oferece características de engenharia de software como o encapsulamento de dados, ocultação de informações, sobrecarga e o tratamento de exceções para os programadores. É a técnica mais fortemente utilizada para o desenvolvimento de aplicações em Oracle

Cada banco de dados tem um nome único geral, fornecido por um vetor hierárquico de nomes de domínio de rede que é colocado como prefixo para o nome do banco de dados para torná-lo único. Os dados em um SGBDD Oracle podem ser replicados utilizando-se *snapshots* [4] (instantâneos) ou tabelas-mestras replicadas. A replicação é fornecida nos seguintes níveis:

Replicação básica: réplicas de tabelas são gerenciadas para acessos *read-only* (somente leitura). Para atualizações, os dados devem ser acessados em um único *site* principal.

Replicação avançada (simétrica): estende-se além da replicação básica, permitindo que aplicações atualizem réplicas de tabelas ao longo de todo um SBDD replicado. Os dados podem ser atualizados em qualquer *site*. Isso requer um *software* adicional chamado *snapshot* como opção de replicação avançada do Oracle.

## 2.8 Banco de Dados Heterogêneos em Oracle

Em um SBDD heterogêneo, pelo menos um banco de dados é um sistema não-Oracle. A ferramenta *Open Gateways* fornece acesso a um banco de dados não-Oracle a partir de um servidor Oracle que utiliza um *link* de banco de dados para

[4] *Snapshots* é uma forma de replicação de uma determinada tabela, onde tem-se uma tabela *master* e réplicas espalhadas em diversos sites. Todas as alterações feitas nas réplicas são efetivadas na tabela *master*.

acessar dados ou para executar procedimentos remotos no sistema não-Oracle. Entre as características do *Open Gateways* incluem-se as seguintes:

Transações distribuídas: sob o mecanismo de *commit* de duas fases, as transações podem transpor sistemas Oracle e não-Oracle.

Acesso transparente da SQL: instruções da SQL emitidas por uma aplicação são transparentemente transformadas em instruções da SQL compreendidas pelo sistema não-Oracle.

Passagem direta da SQL e procedimentos armazenados: uma aplicação pode acessar diretamente um sistema não-Oracle utilizando uma versão da SQL do sistema.

Otimização global de consultas: informações sobre cardinalidade, índices etc, no sistema Oracle, são de responsabilidade do otimizador de consultas do Servidor Oracle para realizar a otimização global de consultas.

Acesso procedural: sistemas procedurais como sistemas de mensagens ou de enfileiramento são acessados pelo servidor Oracle utilizando-se chamadas de procedimentos remotos PL/SQL.

## CAPÍTULO 3 O PROTOCOLO WAP

Por tratar-se de um projeto extenso e que abrange áreas com inovações tecnológicas constantes, este trabalho baseou-se quase totalmente em conteúdos retirados de *sites* da Internet, especializados nas áreas envolvidas. O WAP, por exemplo, foi criado pelo WAPForum no intuito de resolver o problema das limitações que envolvem aplicações sem fio, podendo ser portanto, considerado um manual on-line desta tecnologia. Então, basicamente todo o conteúdo teórico sobre WAP aqui apresentado, baseou-se em informações retiradas das especificações do WAPForum.

O WAP, acesso sem fio (*wireless*) à Internet através de celulares e aparelhos portáteis, tem se tornado cada vez mais evidente em nosso dia-a-dia. Estudos demonstram que em poucos anos, o WAP se tornará o principal canal de acesso à rede mundial de computadores, superando o número de usuários que a acessam via computadores pessoais (tal fato já é realidade no Japão). Diante disso, é impossível ignorar o seu valor estratégico e comercial [Spo02].

O protocolo WAP é basicamente uma pilha de protocolos de comunicação com objetivo de unir um servidor de aplicação a um dispositivo sem fio (*wireless user agent*), numa filosofia Cliente/Servidor, ou seja, o

dispositivo sem fio faz a requisição de alguma informação a um servidor e este lhe responde os dados requeridos. Essa tecnologia especifica dois elementos essenciais para a comunicação sem fio: um protocolo de comunicação fim-a-fim e um ambiente de aplicação baseado em *browsers*. É importante ressaltar que muito da arquitetura WAP foi inspirada na Internet atual, a fim de que o menor esforço possível fosse despendido para o desenvolvimento de aplicações.

## 3.1 O Modelo WWW

A arquitetura Internet provê um modelo de programação flexível e poderoso, em que aplicações e conteúdos são conhecidos como *web browsers* ou navegadores *web*. O *web browser* é uma aplicação para rede, o que significa que ele envia requisições para objetos de dados nomeados para um servidor de rede e este responde com o dado codificado, usando os formatos padrões. Todo o processo descrito é ilustrado na Figura 3. Vale ressaltar que todas as informações do Capítulo 3 foram extraídas de [WF02].

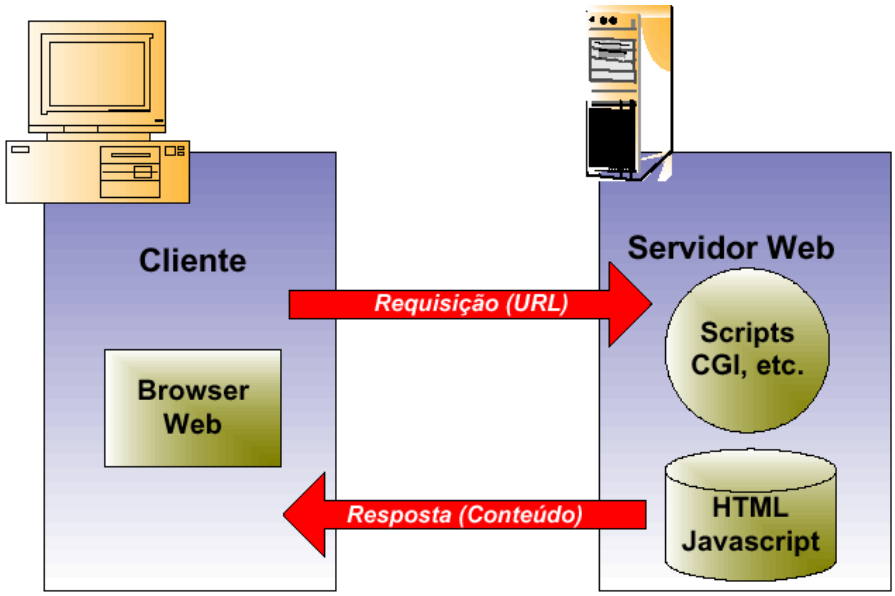



**Figura 3 –** Modelo de Programação *Internet*

Os padrões WWW especificam mecanismo para construção de um ambiente de aplicação, incluindo:

- URL – modelo de nomeação padrão para servidores e conteúdos *Web.*
- Formatos de conteúdos padrão (por exemplo, HTML, *JavaScript*, etc), para que todos os *Web browsers* os suportem.
- Protocolos de comunicação (tais como, HTTP) que permitem ao navegador se comunicar com o servidor.
- Servidor de Origem é onde um dado recurso reside ou é criado.
- *Proxy* é um programa intermediário entre os clientes e servidores, agindo tanto como cliente, quanto como servidor.

- *Gateway* é um servidor que age como intermediário para algum outro servidor.

### 3.2 – O Modelo WAP

O modelo de programação WAP é similar ao modelo de programação WWW. Entretanto, otimizações e extensões foram feitas de maneira que a característica do mundo *web* fosse ao encontro do ambiente sem fio. O conteúdo é transportado usando um conjunto de protocolos padrões de comunicação baseados nos protocolos de comunicação WWW. Um *microbrowser* em terminais sem fio coordena a interface de maneira similar aos *browsers* padrões. Todo o processo é ilustrado na Figura 4.

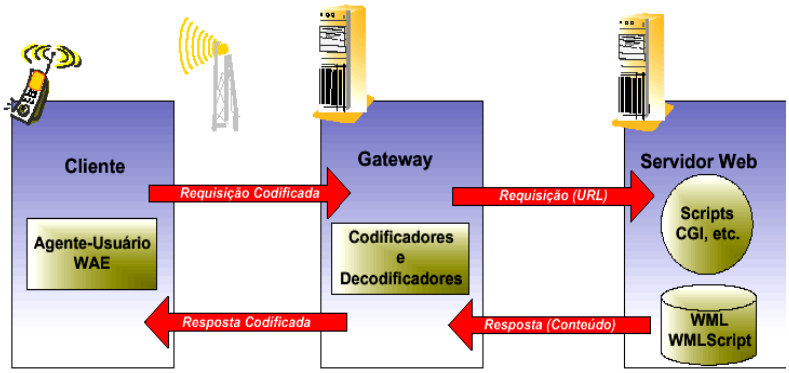



**Figura 4 –** Modelo de programação WAP

O WAP define um conjunto de componentes padrões que permitem a comunicação entre terminais móveis e servidores de rede, incluindo:

- O padrão WWW de URL é usado para identificar o conteúdo WAP nos servidores originais, enquanto o padrão WWW de URI é usado para identificar recursos locais em um dispositivo.
- O protocolo de comunicação WAP possibilita a comunicação do *browser* do terminal móvel com o servidor *web.*
- O *Proxy* WAP conecta um domínio sem fio à *Internet* e tem as seguintes funcionalidades: *Gateway* de Protocolo que traduz as requisições da pilha do protocolo WAP (WSP, WTP, WTLS e WDP), para a pilha do protocolo WWW (HTTP e TCP/IP); permite que conteúdo e aplicações sejam hospedados em servidores WWW e sejam desenvolvidos usando tecnologias WWW como, por exemplo, *scripts* [5] CGI (*Commom Gateway Interface*).
- Codificadores de conteúdo traduzem o conteúdo WAP em formato codificado para reduzir o tamanho dos dados que trafegam pela rede.
- Decodificadores de conteúdo traduzem o conteúdo codificado para o conteúdo WAP.

Na Figura 5, o cliente WAP se comunica com dois servidores da rede sem fio. O *Proxy* WAP traduz as requisições WAP para as WWW, permitindo,

[5] *Script* é a ferramenta responsável por fazer a comunicação entre o banco de dados e o servidor *Web.* Pode ser um programa compilado como o CGI ou interpretado como o PHP e o ASP. Entre os vários outros *scripts Web* para geração de páginas dinâmicas, tem-se também o JSP (*JavaServer Pages*), JavaScript, SSI (*Server-Side Includes*), API (*Application Programming Interface*)

assim, que o cliente WAP possa enviá-las para o servidor *web*. O *Proxy* também codifica as respostas do servidor *web* em um formato binário compacto, que é entendido pelo cliente. Se o servidor *web* provê conteúdo WAP, o *Proxy* o recupera diretamente do servidor, caso contrário um filtro é usado para traduzir o conteúdo WWW para o WAP. Já o servidor WTA é um exemplo de servidor de origem ou *gateway* que responde às requisições do cliente diretamente. O servidor WTA é usado pra prover acesso WAP às capacidades da infra-estrutura de telecomunicações do provedor da rede sem fio.

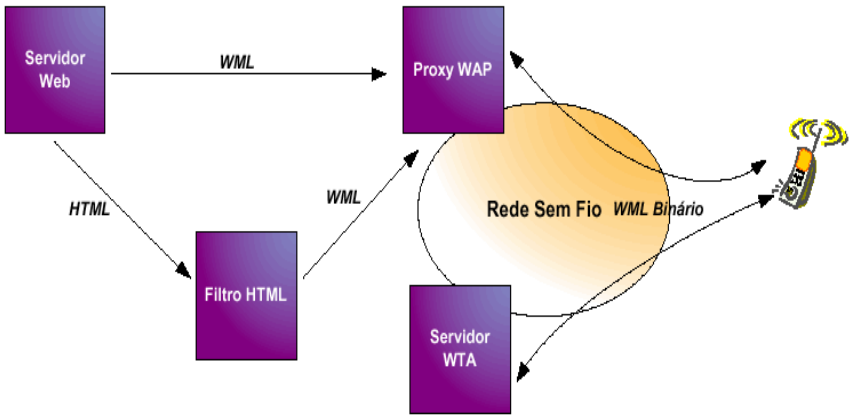



**Figura 5 –** Exemplo de uma rede WAP

A arquitetura WAP provê um ambiente escalável e extensível para desenvolvimento de aplicações para dispositivos móveis de comunicação. Cada camada da arquitetura é acessível pelas camadas abaixo, assim como por outros serviços e aplicações.

A arquitetura em camadas do WAP possibilita outros serviços e aplicações que utilizam as capacidades da pilha WAP através de um conjunto de interfaces bem definidas. Aplicações externas podem acessar as camadas de sessão, transação, segurança e transporte diretamente, permitindo que a pilha WAP seja usada por aplicações e serviços para os quais o WAP não foi inicialmente projetado, mas que são demandados pelo mercado sem fio. A Figura 6 representa uma estrutura em camadas de toda a pilha do protocolo.

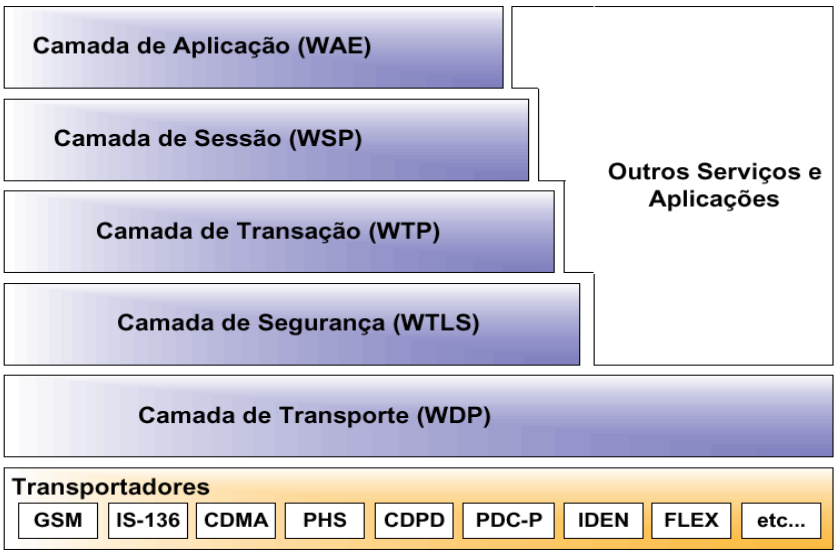



**Figura 6** – A Arquitetura WAP

3.2.1 - WDP (*Wireless Datagram Protocol*)

O WDP compõe a camada de transporte do WAP e opera acima dos serviços de transporte de dados suportados pelo vários tipos de redes. Por ser um datagrama de serviço de caráter geral, oferece um serviço consistente para os protocolos das camadas superiores (Segurança, Transação e Sessão) e se comunica de maneira transparente com um dos serviços transportadores disponíveis. Na Figura 7, as áreas acinzentadas são as camadas do protocolo onde a especificação do WDP é aplicável.

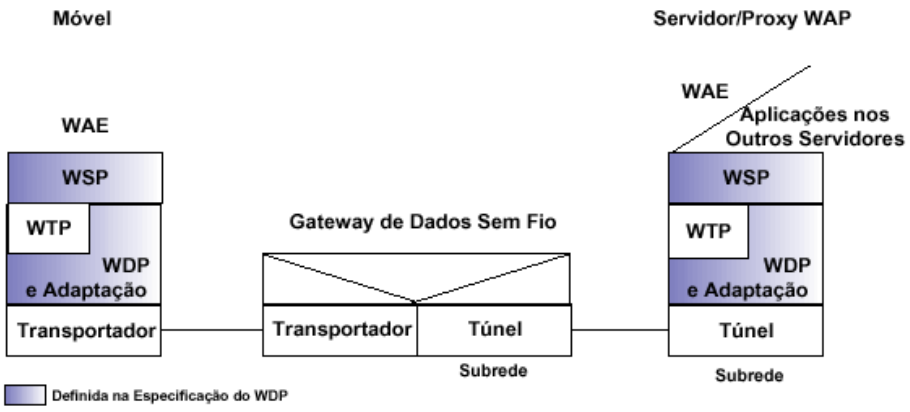



**Figura 7 -** Arquitetura Genérica do WDP

Os serviços oferecidos pelo WDP incluem endereçamento de aplicação por número de porta e segmentação opcional, além de reunião e detecção de erros opcionais. Estes serviços permitem às aplicações operarem transparentemente sobre diferentes transportadores disponíveis.

### 3.2.2 - WTLS (*Wireless Transport Layer Security*)

É um protocolo de segurança baseado no protocolo padrão da indústria, o TLS, formalmente conhecido como SSL. O objetivo primordial da camada WTLS é fornecer privacidade, integridade de dados e autenticação entre duas aplicações. O protocolo da Camada de Segurança na arquitetura WAP provê para as camadas superiores um serviço de interface de transporte seguro, preservando o serviço de interface de transporte abaixo dele.

A camada WTLS é modular, isto é, o seu uso ou não depende do nível de segurança requerido em uma dada aplicação. Ou seja, é uma camada opcional da pilha WAP. E além da segurança, o WTLS fornece uma interface para gerenciamento de conexões seguras.

O gerenciamento de conexão do WTLS permite ao cliente conectar-se com um servidor e concordar com as opções do protocolo a serem usadas. O estabelecimento de uma conexão segura consiste de uma série de passos e tanto o cliente quanto o servidor podem interromper a negociação de acordo com sua vontade. A negociação pode incluir os parâmetros de segurança, como algoritmos de criptografia, tamanho de chaves, troca de chave e autenticação.

### 3.2.3 - WTP (*Wireless Transaction Protocol*)

WTP roda no topo do serviço de datagrama e oferece benefícios como a melhoria da segurança sobre os serviços de datagrama, o que libera a camada

superior de retransmissões e reconhecimentos necessários quando os serviços datagrama são usados. Outro benefício é a melhoria da eficiência dos serviços orientados à conexão.

O WTP é orientado a mensagens e é projetado para serviços orientados a transações, como navegação. O protocolo possui várias características:

- Transações de classe 0 – uma aplicação que requer serviço "*push* não confiável" provê um serviço de datagrama não confiável.
- Transações de classe 1 – uma aplicação que requer serviço "*push* confiável" provê um serviço de datagrama confiável.
- Transações de classe 2 – uma aplicação que requer serviço básico de execução / resposta. Uma sessão WSP pode consistir de uma série de transações deste tipo.

O WTP possui uma entidade de gerenciamento que monitora o estado sob as capacidades e serviços do ambiente do dispositivo move e notifica a camada WTP se um ou mais serviços assumidos na estão disponíveis. Por exemplo, se o dispositivo sai da área de cobertura do transportador, bateria baixa, monitoramento do estado do ambiente móvel (endereço do dispositivo). A entidade de gerenciamento trata todos os assuntos relativos a inicialização, configuração, reconfiguração dinâmica e recursos de camada WTP.

### 3.2.4 - WSP (*Wireless Session Protocol*)

O WSP provê à camada de aplicação uma interface consistente entre duas sessões de serviço, onde a primeira é um serviço orientado à conexão acima do protocolo da camada de transação (WTP) e a segunda é um serviço sem conexão que opera acima de um serviço de datagrama seguro ou não.

Especificamente, ele fornece às aplicações meios para estabelecer uma sessão confiável do cliente ao servidor, liberar esta sessão de maneira ordenada e concordarem com um nível comum da funcionalidade do protocolo usando capacidade de negociação, bem como trocarem conteúdo entre cliente e servidor usando codificação compacta, além de fornecer meios para suspender e continuar a sessão.

O WSP provê transferência *push* e *pull* de dados, onde o *pull* é feito usando mecanismo de requisição e resposta do HTTP/1.1. Para transferência *push,* o WSP fornece três mecanismos de transferência de dados:

1. *Push* de dados confirmado dentro do contexto de uma sessão existente – permite ao servidor enviar dados para o cliente a qualquer momento durante a sessão e o servidor recebe a confirmação de que o *push* foi entregue.
2. *Push* de dados não-confirmado dentro do contexto de uma sessão existente – similar ao descrito sobre *push* de dados confiável, mas sem confirmação de recebimento.

3. *Push* de dados não-confirmado sem o do contexto de uma sessão existente – o contexto padrão de uma sessão é assumido. *Push* de dados não-confirmado podem ser usados para enviar mensagem em um único sentido, sobre um transporte não confiável.

Com a funcionalidade *push,* os usuários são capazes de receber informações sem ter que requisitá-las, por exemplo, previsões de tempo, notificações de eventos, como chegada de *emails*, dentre outros.

### 3.2.5 - WAE (*Wireless Application Protocol*)

O WAE é a camada responsável por estabelecer um ambiente interativo onde operadoras e prestadores de serviços poderão construir aplicações que alcançarão um grande número de plataformas sem fio de uma maneira eficiente.

O WAE inclui um ambiente de micro-*browser* contendo as seguintes funcionalidades:

- WML(*Wireless Markup Language*), uma linguagem de marcação leve, similar ao HTML(*HyperText Markup Language*), mas otimizada para utilização em dispositivos móveis:
- WMLScript, uma linguagem de script leve, similar ao JavaScript;

- Serviços de telefonia e interfaces de programação: *Wireless Telephony Application* (WTA, WTAI);
- Um conjunto de formatos de dados bem definidos, incluindo imagens, registros de agenda de telefones e de compromissos.

O WAE assume a existência de funcionalidades de *gateway* responsáveis pela codificação dos dados transferidos para o cliente móvel. O objetivo da codificação dos dados é minimizar o tamanho desses dados enviados por ar, e a utilização de recursos necessário para o cliente processar esses dados. A funcionalidade do *gateway* pode ser adicionada a servidores já existentes ou colocada em *gateways* dedicados. A funcionalidade de *gateway* pode ser adicionada aos servidores de origem ou ser colocada em *gateways* dedicados, conforme Figura 8.

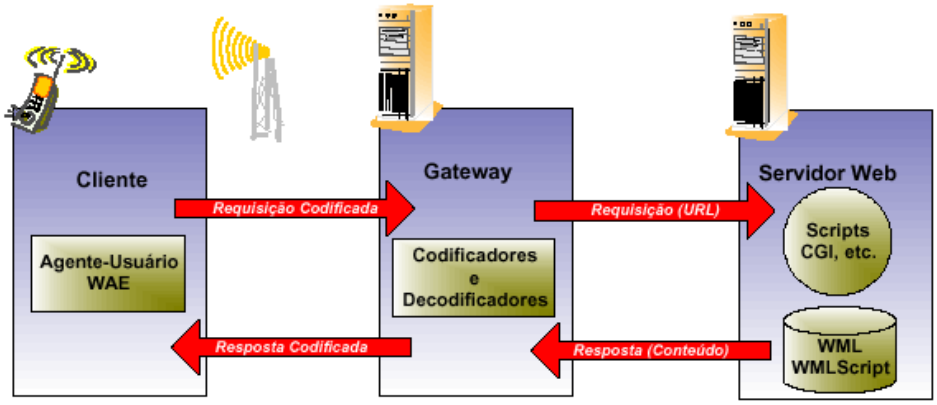



**Figura 8** - Modelo lógico do WAE

O resultado da arquitetura WAE, será um modelo que oferece as seguintes vantagens:

- Tira vantagem dos padrões, tecnologia e infra-estrutura desenvolvidas para a Internet;
- Disponibiliza serviços de rede móvel avançados ao usuário, através de serviços de telefonia controlados pelo operador da rede;
- Fornece uma área de trabalho extensa para a construção de serviços *wireless*.

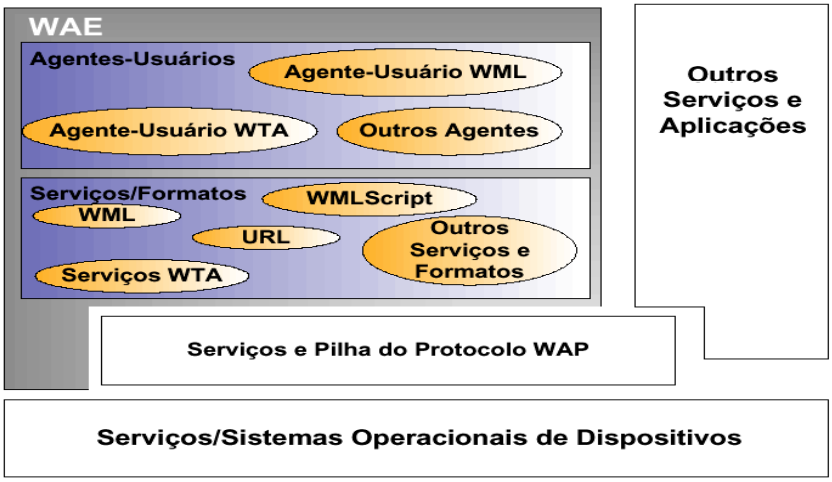



**Figura 9 –** Componentes do Modelo lógico WAE

Os componentes do modelo lógico WAE são mostrados na Figura 9 e descritos nos itens abaixo:

1) Agentes-Usuários

O WAE não especifica formalmente nenhum agente-usuário, ele somente define serviços e formatos fundamentais que são necessários para assegurar a interoperabilidade entre as diferentes implementações.

2) Serviços e formatos

WML

O WML é uma linguagem de marcação baseada no XML e objetiva ser usada na especificação de conteúdo e interface para dispositivos de banda estreita, incluindo telefones celulares e *pagers*. O WML e seu ambiente de suporte foram projetados tendo em vista as restrições dos dispositivos de banda estreita, o que inclui pequenos *displays*, facilidades de entrada do usuário limitadas, conexões para rede de banda estreita e recursos de memória e processamento limitados.

O usuário navega em um conjunto de *cards* WML, os quais contém instruções que podem invocar serviços no servidor. Um conjunto de *cards* forma um *deck*. *Decks* podem ser requisitados ao servidor conforme sejam necessários. Os *decks* WML podem ser armazenados em arquivos estáticos ou podem ser gerados dinamicamente por geradores de conteúdo como, por exemplo, *scripts* ou *servlets*.

WMLScript

É uma linguagem de *script* leve e procedural, que realça as facilidades de navegação e apresentação do WML, além de possuir: capacidade de suporte a comportamento da UI *(User Interface)* de modo mais avançado; capacidade de adicionar inteligência ao cliente; capacidade de prover um mecanismo conveniente de acesso aos dispositivos e seus periféricos; e reduzir a necessidade de idas e voltas ao servidor.

URL

O WAE utiliza bastante a semântica do HTTP(*Hypertext Markup Protoco*l ) para definição de endereços e, em alguns casos, seus componentes estendem a da URL como no WML, em que os fragmentos de uma URL podem ser usados para permitir a ligação seja uma chamada para funções particulares do WML*Script*.

Formatos do Conteúdo

O WAE inclui um conjunto de formatos do conteúdo, que facilitam a troca de dados, entretanto o método de troca irá depender dos dados e do agente-usuário alvo do WAE. Os dois formatos mais importantes na camada são os de *bytecode* codificados do WML e do WML*Script*, que tornam a transmissão destes mais eficiente e minimiza a necessidade de

capacidade computacional do cliente. Além destes, o WAE adota outros formatos de dos como imagens, mensagem e formatos específicos do agente-usuário.

3) Transporte do WML e do WML*Script*

Os servidores de origem provêem os serviços de aplicação para o usuário final e a interação destes serviços entre o usuário final e os servidores de origem é empacotada em *decks* WML e *scripts* padrões, os quais podem estar armazenados estaticamente ou serem dinamicamente gerados. Um codificador WML em um *gateway* converte cada *deck* WML em seu formato binário, que depois é enviado ao cliente para ser interpretado e mostrado.

A existência de um *gateway* não é obrigatória, uma vez que a compilação e codificação podem ser funções internas dos servidores de origem, o que não afetaria em nada o processo descrito. A Figura 10, mostra a arquitetura lógica do agente-usuário do WML.

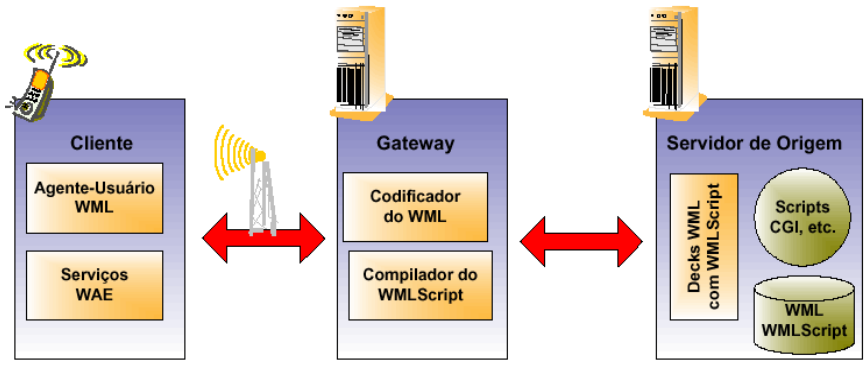



**Figura 10** – A arquitetura lógica do agente-usuário do WML.

E, a seguir, a figura 11 ilustra a arquitetura lógica sem um *gateway* agente-usuário.

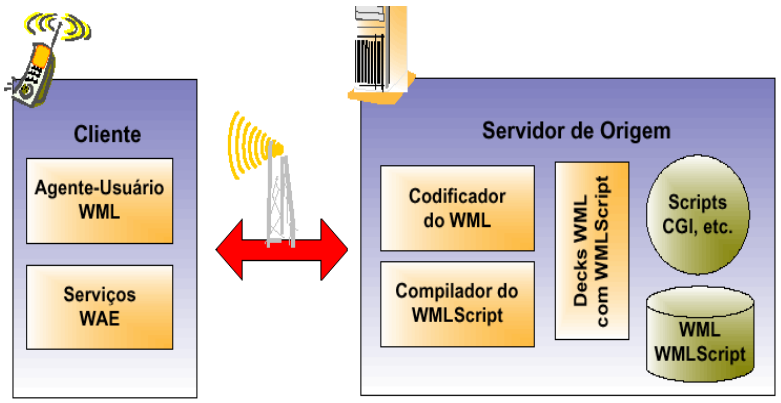



**Figura 11 –** Arquitetura lógica sem o agente-usuário WML

4) WTA (*Wireless Telephony Application*)

Uma possível configuração da estrutura da WTA é ilustrada na Figura 12. Essa ilustração leva em conta somente os componentes do cliente. Ou seja, como o agente-usuário WTA, o repositório e a WTAI (*Wireless Telephony Application Inteface*) interagem uns com os outros e com as entidades em um dispositivo cliente móvel com capacidades de usar a WTA.

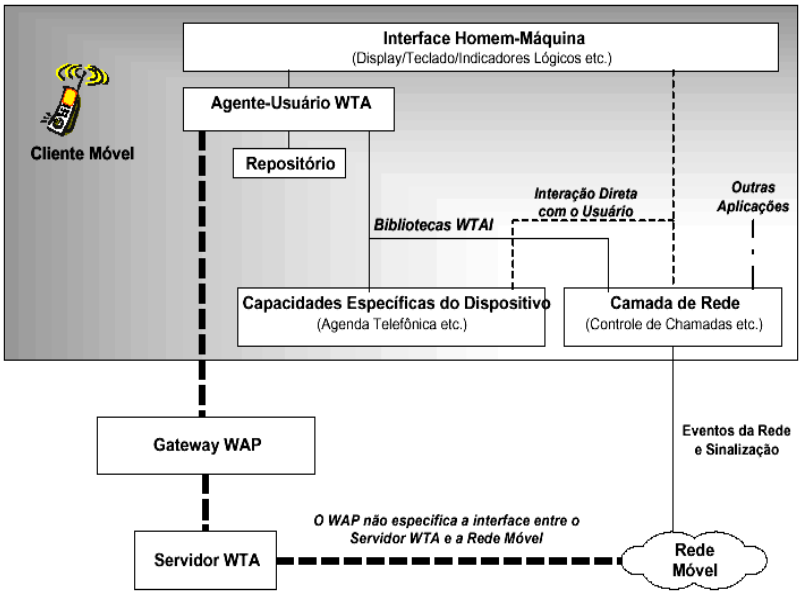



**Figura 12 –** Exemplo da Arquitetura WTA

Os Agentes-Usuários WTA e WAE

Para suportar funções de telefonia de dentro de um agente-usuário do WAE, é muito importante uma biblioteca especial do WTA: a biblioteca

Publica WTAI, que contém funções que podem ser chamadas de dentro de qualquer aplicação WAE, de acordo com a Figura 13, ilustra como o agente-usuário WAE e a Biblioteca Pública WTAI interagem uma com o outro e com outras entidades de cliente móvel.

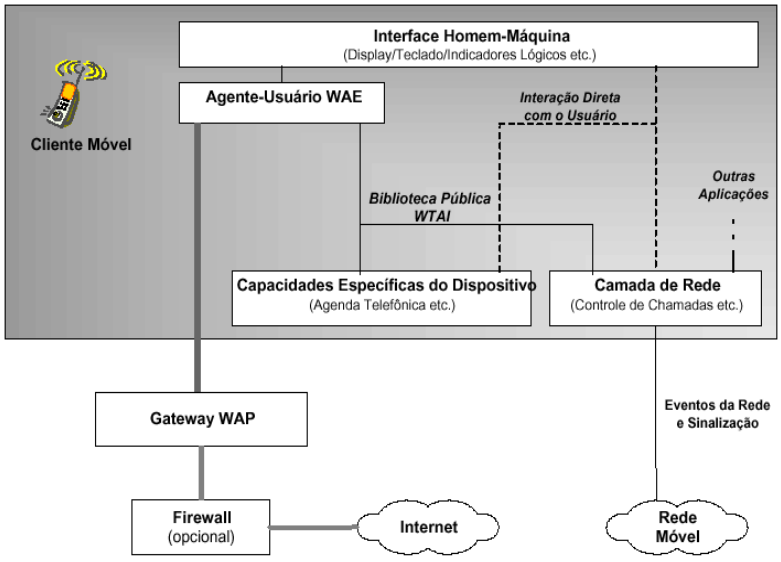



**Figura 13 –** O agente-usuário WTA e a Biblioteca Pública WTAI

# CAPÍTULO 4 BANCO DE DADOS DISTRIBUÍDOS E REDES WIRELESS

As características do Oracle9*i Application Server (AS)* fornecem uma infra-estrutura móvel, com multi-canais de acesso a qualquer aplicação, incluindo voz. Desde já, vale ressaltar que todo o conteúdo deste Capítulo foi retirado de [Ora02]. O componente Oracle9*iAS Wireless* existente nesta infra-estrutura é a plataforma de desenvolvimento e disponibilização de aplicações móveis. A aplicação desenvolvida é independente da infra-estrutura sem fio, incluindo redes, protocolos, dispositivos, linguagens de marcação e *gateways*, conforme mostra a Figura 14.

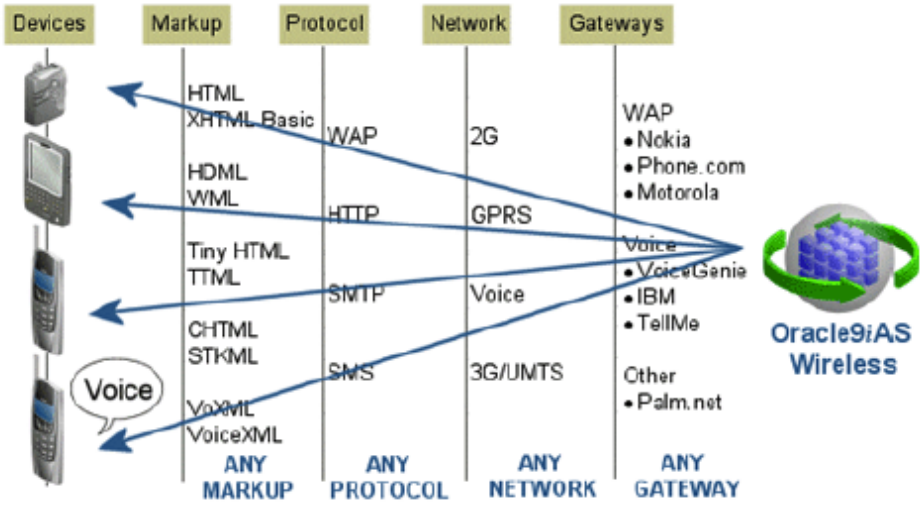



**Figura 14** – Oracle9iAS *Wireless*

Como podemos observar é possível que uma única aplicação suporta qualquer tipo de dispositivo, qualquer tipo de linguagem de marcação, qualquer protocolo, qualquer tipo de rede e qualquer tipo de gateway.

## 4.1. Infra-estrutura *Wireless*

Existem muitos componentes estruturais que interoperam na rede sem fio. Estes componentes típicos estão descritos na Figura 15.

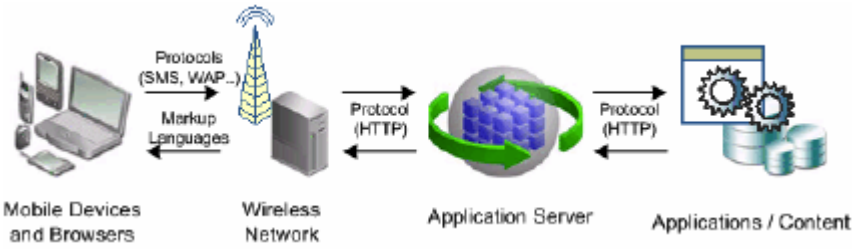



**Figura 15** - *Wireless infrastructure components*

A seguir cada um dos componentes ilustrados na Figura 15 será descrito detalhadamente.

*Mobile Devices and Browsers* (Dispositivos Móveis e Browsers)

São usados para acessar a Internet móvel. Todo dispositivo móvel geralmente usa um *browser* para mostrar informações recebidas. A linguagem de marcação (aquela que especifica como a informação deve ser mostrada na tela do

dispositivo) usada por dispositivos móveis é uma linguagem que interpreta o *microbrowser* do dispositivo *wireless.* Como exemplos temos o *Voice*XML, WML e HDML.

*Wireless Network* (Rede Sem Fio)

Uma importante característica de redes é a largura de banda e o tipo de conexão. Por exemplo, geração 2,5G e 3rd fornecem uma alta velocidade de transmissão de dados.

Protocolos são usados para distribuir o conteúdo aos dispositivos. Exemplos de protocolos são o WAP, voz e SMS. Note que protocolos sem fio são mais eficientes sobre a rede sem fio que o protocolo padrão HTTP. Esta é uma das principais razões pelas quais muitos clientes da Internet sem fio não podem acessar o protocolo *web* HTTP diretamente. *Wireless Gateways* traduzem a requisição feita no protocolo sem fio para o protocolo padrão HTTP da *web*. *Gateways* traduzem uma variedade de protocolos tais como WAP, SMS, voz e outros.

*Application Server and Appllications / Content* (Servidores de Aplicação e Aplicações e Conteúdo)

Servidores de aplicação são usados para aumentar a eficiência de desenvolvimento, disponibilização e gerenciamento. Um servidor de aplicação

sem fio associa o "conteúdo-fonte" de uma requisição sem fio ao padrão de *Gateway wireless* da rede sem fio. Feito isso, ele recupera o conteúdo originado do "conteúdo-fonte", personaliza ao seu usuário, e o transforma na linguagem de marcação específica compreendida pelo dispositivo que está sendo usado. Isso tira dos desenvolvedores a responsabilidade de projetar todos protocolos e marcadores durante o desenvolvimento. Além disso, servidores de aplicação incluem ferramentas (teste, controladores de dispositivos etc) e *frameworks* (*messaging, offline, provisioning*, entre outros) o que simplifica o processo de desenvolvimento.

Aplicações / Conteúdo têm uma enorme variedade de formas incluindo informações de bancos de dados, personalização de conteúdos, alertas, e-mail, serviços locais etc. A grande variedade de tipos de conteúdos aumenta a complexidade de encontrar uma forma gerenciável para distribuir cada aplicação em todo tipo de dispositivo da melhor forma possível.

### 4.1.1. O Processo de Requisição Móvel

Esta seção descreve o processo realizado quando um dispositivo móvel solicita um serviço ao servidor de aplicação *wireless* na rede. A figura 16 ilustra o processo de requisição móvel.

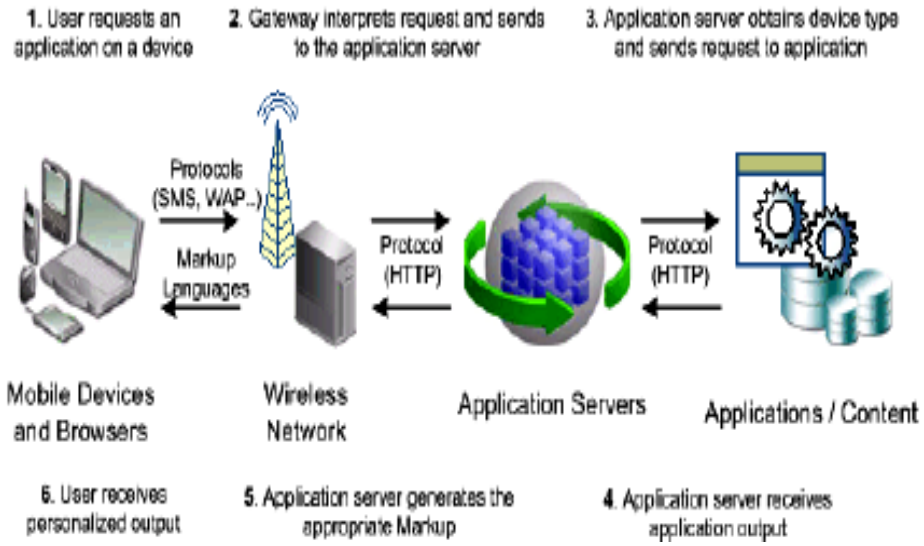



**Figura 16** – *Wireless request process*

A seguir cada um dos passos necessários para a realização de uma solicitação móvel é descrito.

Envio de requisição

Um usuário chama um serviço usando um dispositivo *wireless.* Depois, o usuário disca o número do telefone para o provedor de serviços apropriado – em redes 2.5G e 3G de acesso *always-on* não é necessário discar um provedor de serviços para iniciar uma sessão. O *microbrowser* do dispositivo *wireless* envia um pedido para a estação rádio base da rede sem fio. O pedido pode ser enviado sobre uma variedade de protocolos diferentes, tais como SMS ou WAP,

dependendo do tipo do dispositivo usado. Note que todos estes protocolos foram otimizados para funcionar sobre uma rede *wireless* com largura de banda limitada e conectividade intermitente. *Packet-based protocols* são mais eficientes sobre a rede *wireless* existente que o protocolo de *Internet* padrão, o HTTP.

Reconhecimento e autenticação do dispositivo *Wireless*

Quando a estação rádio base da rede sem fio recebe o pedido, ela solicita ao dispositivo móvel que ele identifique a ordem para proceder com a autenticação. Depois de estabelecida uma sessão, o WAP *gateway* passa a informação sobre a solicitação *web* para o servidor de aplicação *wireless.* O cabeçalho da mensagem contém informação tais como identificação do usuário, o dispositivo que o usuário está acessando a *Internet*, a localização geográfica do usuário, e o endereço ou serviço que o usuário está acessando. Esta informação será usada pelo servidor de aplicação *wireless* para personalizar a interação com o usuário.

Estabelecimento de uma sessão *Wireless*

Uma vez bem sucedida a autenticação, o provedor de serviços aceita a chamada e estabelece a conexão com o dispositivo móvel. O pedido é enviado da estação rádio base sobre a rede sem fio usando o *Wireless Transport Protocol (WTP)*. O operador do *gateways wireless* recebe o pedido.

Tradução do pedido para o padrão internet

Note que ainda que o protocolo padrão da rede celular não é o mesmo protocolo padrão da Internet, portanto ele precisa ser convertido para o protocolo padrão da Internet (o HTTP), depois o pedido é passado da rede sem fio para a tradicional Internet – um *gateway* fornece esta função. Para que dispositivos WAP sejam capazes de transpor o WTP para o HTTP é necessário um WAP *Gateway*. O WAP *Gateway* não só codifica a mensagem de um protocolo para outro, mas também reconhece como passar a mensagem da rede sem fio para a infra-estrutura tradicional da Internet.

Conexão com o servidor de aplicação

O *gateway* converte o pedido para uma URL, para um *web site* específico. Feita a conversão, a mensagem é enviada para o servidor de aplicação *wireless,* como um pedido padrão da Internet – uma URL específica ou um endereço a ser acessado. O servidor de aplicação e o *gateway,* através do processo de autenticação, autenticam cada mensagem e estabelecem a sessão.

Reconhecimento da informação sobre o usuário

O servidor de aplicação interpreta a identificação do usuário, o dispositivo, a localização e outros parâmetros que são passados para ele.

Processamento da solicitação *Wireless*

Quando o servidor de aplicação *wireless* recebe o pedido, ele o processa em três passos: a) Recupera o conteúdo que a aplicação *wireless* está sendo acessado, b) Adapta o conteúdo para o usuário, e c) Transforma o conteúdo para o padrão específico do dispositivo que está sendo usado. A adaptação do conteúdo é feita agregando o conteúdo da aplicação acessada para o formato XML

Customizando o conteúdo para algum usuário

Oracle9iAS Wireless também reconhece o contexto da sessão do usuário e customiza os serviços interpretando para o usuário final. Oracle9iAS Wireless permite aos usuários que eles mesmo customizem seu portal escolhendo quais serviços eles gostariam de ver, organizando serviços de notificação e personalizando serviços de acordo com tipo do dispositivo que usou para acessar a Internet e com sua posição geográfica (Location-based Service).

Adaptação do conteúdo para o dispositivo / rede apropridado

Finalmente, já que cada usuário pode usar um ou mais dispositivos diferentes para acessar a Internet e cada dispositivo reconhece uma linguagem de marcação diferente, Oracle9iAS Wireless transforma o conteúdo, renderizando-o para a linguagem de marcação reconhecida pelo dispositivo que está sendo usado. Note que a maioria dos servidores de aplicação pode somente adaptar um número

muito limitado de conteúdos e eles podem renderizá-los para um pequeno número de dispositivos (normalmente, somente aqueles que entendem WAP/WML e conteúdo HDML). Oracle9iAS Wireless pode ser um servidor WAP, mas mais significativamente ele usa XML para traduzir o conteúdo para o formato compatível com o dispositivo usado.

### 4.1.2. Limitações no Desenvolvimento de Aplicações Móveis

Dificuldade em realizar a entrada de dados em dispositivos móveis

Mesmo dispositivos móveis com uma interface mais amigável, como um Palm e iPaq, são muito mais complicados de usar que um PC com mouse e teclado. Limitações deste tipo dificultam como a Internet pode ser acessada de um dispositivo móvel.

Tamanho Limitado da Tela do Dispositivo

O tamanho da tela e a capacidade de armazenamento dos dispositivos variam muito. E isto faz com que a plataforma deva ser capaz de explorar funcionalidades de dispositivos específicos, tais como voice brownsing as quais permitem navegar facilmente por aplicações de um dispositivo wireless. Além disso, a plataforma deve suportar uma variedade de serviços personalizados, tais como permissão para usuários personalizarem quais serviços eles quiserem,

obtenção de informações sobre serviços baseadas no dispositivo que estiverem usando, e obtenção e acesso a informações de acordo com a localização geográfica.

Fonte de Conteúdos Heterogêneos

A plataforma de aplicação wireless deve suportar padrões abertos de desenvolvimento de aplicações, tais como, Java e XML O servidor wireless também deve fornecer uma estrutura com funções móveis pré-definidas como location-awareness, customization e messaging. A plataforma de aplicação wireless deve suportar conteúdos de diversas fontes, tais como, de um web site, de um servidor de e-mail, ou de um banco de dados. Além disso, deverá suportar também qualquer aplicação seja ela construída em Java, Visual Basic, PERL, PL/SQL, PHP etc, como mostra a Figura 17.

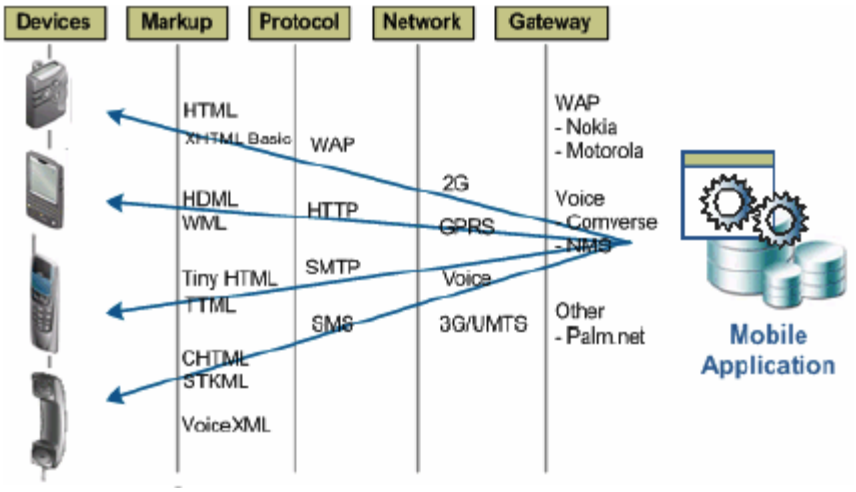



**Figura 17 –** *Complications of móbile application development*

Desempenho da Aplicação e Escalabilidade

Uma plataforma Internet *Wireless* deve suportar e ser capaz de gerenciar um grande número de usuários. Deve garantir suas seguranças e controle de acesso a privilégios. Uma plataforma Internet *Wireless* deve ser capaz de gerenciar sessões de usuários e preservar a sessão em uma alta faixa de escalabilidade. Uma plataforma Internet *Wireless* deve ser capaz de usar um *cache* e *share data* para agilizar a recuperação de grandes volumes de conteúdos.

Necessidade de desenvolvimento de um Wireless Internet Market

A maioria das plataformas de aplicação *wireless* atende a um pequeno conjunto de necessidades, pedindo ao usuário que escolha uma infra-estrutura de software

para *messaging,* outra pra buscar conteúdo, outra pra comércio móvel, outra pra acesso por voz, etc. Assim, é preciso que uma plataforma integre essas necessidades a fim de permitir o desenvolvimento de aplicações *state-of-the-art* e portais.

Evolução de Padrões *Wireless*

Tecnologias de transmissão de dados tais como, CDMA (*Code Division Multiple Access*), GSM (*Global System for Mobile Communication*), TDMA(), iDEN, SMS(), i-Mode, GPRS(), e UMTS() estão todos se desenvolvendo rapidamente. Padrões de dispositivos como VoiceXML prometem mudar como a *Internet Wireless* é usada. Em conseqüência disso, uma plataforma de aplicações *Wireless* deve ser compatível com padrões como i-Mode, WAP, SMS, GPRS, 3G, e outros. Deve também suportar padrões abertos como XML, XHTML, *JavaServlets, Java Server Pages* para desenvolvimento de aplicações.

## 4.2. Oracle9*i* como Solução *Wireless*

O Oracle9*i*AS faz o *web site* com todas aplicações acessíveis de qualquer *browser* ou dispositivo móvel. A Figura 22 mostra a arquitetura do Oracle9*i*AS.

Para empreendimentos que procuram por maior eficiência em tecnologia móvel, o componente Oracle9*iAS Wireless* inclui serviços móveis tais como PIM e *email, location-based* e *messagin services* (via SMS, WAP-*push*, fax, *email* e

voz). Estes serviços podem ser estendidos com funções customizadas. Portanto, Oracle9*iAS Wireless* é um padrão aberto com soluções integradas para desenvolvimento aplicações móveis. A Figura 18 mostra a arquitetura do Oracle9*i*AS *Wireless.*

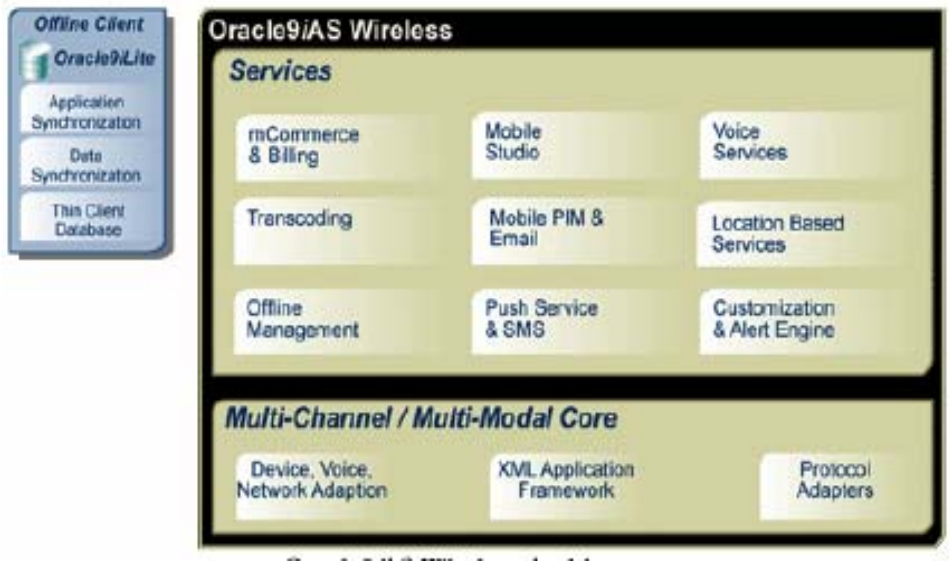



**Figura 18 –** Oracle9*i*AS *Wireless Architecture*

As ferramentas componentes da arquitetura Oracle9*iAS Wireless Architecture* serão descritas nas sessões seguintes.

### 4.2.1. Desenvolvimento *Multi-Channel* (múltipos canais), *Multi-Modal* (múltiplas bandas)

Oracle9*iAS Wireless Multi-Modal* é uma ferramenta que dá aos desenvolvedores o poder de trabalhar independentemente do tipo de rede, protocolo, dispositivos, *gateway* e de outras complexidades que envolvem *wireless.*

Oracle9*iAS Wireless core* normaliza as complexidades para um protocolo e uma linguagem, o HTTP e o XML. A centralização em um Proxy com múltiplos canais permite que desenvolvedores criem uma aplicação somente uma vez suportando vários canais, tais como voz, *push* e SMS.

Para utilizar o *core,* é preciso criar uma aplicação XML e referenciar o Oracle9*i*AS *Wireless core* para a aplicação com uma URL. O Oracle9*i*AS *Wireless core* detectará o dispositivo e renderizará a aplicação apropriadamente, e, ao mesmo tempo, dirá as características específicas do dispositivo.

O HTTP *Adapter* é usado para recuperar conteúdos móveis de todo servidor HTTP / XML. Ele recupera seguramente o conteúdo da aplicação e o entrega ao centro de processamento. O HTTP*Adapter* suporta HTTP1.1 e características específicas como reendereçamento e *cookies.*

A ferramenta de aplicação XML multi-canal é o segredo para o desenvolvimento único da aplicação para múltiplos canais. Essa ferramenta suporta ligações de serviços (*Web Service*), noção de localização, e contexto de

informações para fornecer aos desenvolvedores a capacidade de rapidamente desenvolver aplicações móveis com o máximo de eficiência.

Adaptações em redes e dispositivos transformam e otimizam o conteúdo da aplicação para todo tipo de rede e dispositivo. Cerca de 100 dispositivos móveis, incluindo, *2-way pagers* para serviços assíncronos (SMTP / SMS), todos dispositivos WAP, acesso por voz através de linhas telefônicas comuns, PDA etc. são suportados por essa ferramenta. A Oracle possui um sistema de manutenção e suporte para a maioria dos produtos anteriormente citados.

Oracle9*iAS Wireless* usa o Oracle9*i Database* como repositório para armazenagem de objetos da aplicação. APIs fornecem a funcionalidade de manipular esse armazém de dados. Oracle9*iAS Wireless* APIs podem ser usados para customizar procedimentos, como, fornecimento de diferentes esquemas de autenticação ou customização de mecanismos de identificação de dispositivos. Permite ainda, uma ferramenta de extensão, a qual permite "*plugar*" sistema de monitoramento ou *logging*.

*Voice Network Adaption* ( Disponibilização de Aplicações Via Voz )

Acesso a aplicações via voz implica no usuário chamar um servidor em uma linha telefônica e interagir com ele numa interface de áudio. Há dois métodos de conexão para o usuário: ou por fala ou por *number pad*. Antes de realizar o

reconhecimento de voz, o usuário precisa programar o *software* para reconhecer sua identificação.

Os componentes do processo são três: o *gateway* de voz, o servidor de aplicação e o conteúdo fonte. O servidor de aplicação é responsável por gerar páginas VoiceXML de um Oracle9*i*AS *Wireless* XML. Estas páginas VoiceXML chamam tarefas usando vozes, e podem ser customizadas para chamadas a arquivos pessoais como os armazenados em bancos de dados. A Figura 19 ilustra o processo de requisição por voz.

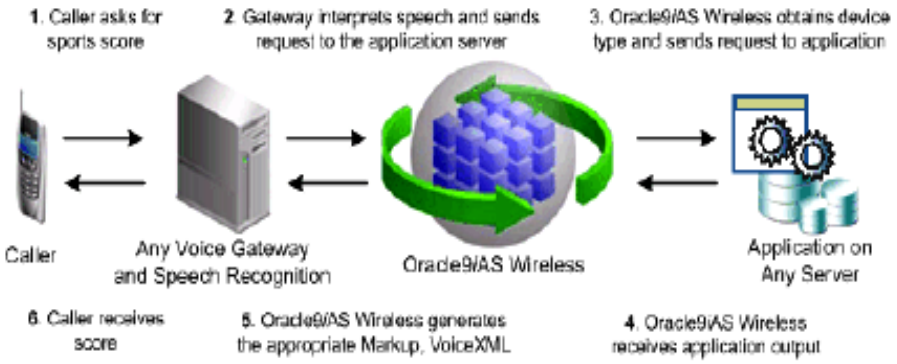



**Figura 19 –** *Voice Request process*

### 4.2.2. *Push Service SMS* (Criação de Aplicações *Push* e SMS)

Oracle9*i*AS *Wireless Push Service* fornece um mecanismo de entrega de mensagens a dispositivos móveis altamente escalável. Essas mensagens podem ser entregues de diversas formas como, SMS, e-mail ou voz. O *Push Service é*

construído em uma arquitetura de entrega de mensagens escalável que suporta um grande volume de mensagens para diferentes tipos de dispositivos. Além disso, fornece também formas de gerenciar e gravar mensagens, incluindo o estado da mensagem entregue.

### 4.2.3. Customização da Interação entre Aplicação e Usuário Final

Aplicações móveis são mais eficientes quanto maior for a satisfação das necessidades do usuário. O Oracle9*i*AS *Wireless alert subscription APIs* ou *o* Oracle9*i*AS *Wireless Content Manager* permitem a construção de aplicações que resultam numa interação *one-to-one* entre o usuário e a aplicação. Usuários finais podem se inscrever ou se desinscrever em tópicos de alertas, de acordo com suas necessidades.

### 4.2.4. Reuso de Aplicações *Web* existentes

Oracle9*i*AS *WirelessTranscoding Service* permite que aplicações que tenham sido desenvolvidas para um dispositivo particular ou em uma linguagem específica sejam transformadas para outro padrão de dispositivo, incluindo voz.

Oracle9*i*AS *Wireless* suporta um serviço de adaptação ou tradução de conteúdo. O conteúdo é transformado para o formato XML do Oracle9*i*AS *Wireless* e então, renderizado para linguagem de marcação padrão do dispositivo.

O WML *translator* entrega aplicações WML (WAP) para dispositivos que não usam o WML como linguagem padrão. O objetivo do serviço WML *transcoding* é fornecer uma forma simples para se quebrar a barreira encontrada no processo de desenvolvimento de aplicações *wireless.* Oracle9*i*AS *Wireless* traduz o WML em XML assim torna-se uma linguagem comum para dispositivos móveis que possuem limitações, como padrões de dispositivos.

### 4.2.5. Gerenciamento *Offline*

É usado em casos onde a conexão móvel não existe ou é baixa. Ele capacita os usuários a usar aplicações sem qualquer rede de acesso. Quando a conexão é reestabelecida o usurário pode se conectar ao servidor para atualizar as novas informações. O Oracle9*i Lite* fornece este tipo de serviço.

### 4.2.6. Noção da Localização das Aplicações

Aplicações *Location-aware* tomam decisões baseadas na localização geográfica do usuário. Esse serviço é conseguido através do Oracle9*i*AS *Wireless Location-Based Service,* e não só reduz o número de entradas e baixa o tempo gasto para obter informação, como também melhora a eficiência, possibilitando acesso a informações relevantes para o usuário, tais como, mapas, caminhos a seguir, tráfego e serviços.

# CAPÍTULO 5 O SISTEMA DE PRONTO ATENDIMENTO REMOTO HOSPITALAR

O objetivo deste trabalho é conciliar estas duas tecnologias, Banco de Dados Distribuídos e a tecnologia WAP, numa aplicação que atenda às necessidades estratégicas, econômicas e sociais de determinada região. Dessa forma, seria possível integrar informações distribuídas em diversas localidades de uma região e acessá-las em qualquer parte através dos dispositivos sem fio. Portanto, a intenção deste projeto é integrar sistemas autônomos e heterogêneos via *Internet e* acessá-los via rede *Wireless.*

Cada um dos hospitais participantes do Sistema de Pronto Atendimento Remoto (SPARH) possui tecnologias e procedimentos operacionais próprios, incluindo o processamento específico dos dados locais. Em função disso, os bancos de dados distribuídos nas aplicações correspondentes apresentam elementos distintos. Assim, devido a essa diversidade, propôs-se a manutenção da autonomia de cada um dos hospitais participantes, então, para alcançar este objetivo, o problema da heterogeneidade foi resolvido no nível conceitual dos bancos de dados. O mapeamento entre esquemas conceituais permitiu uma visão global e complementar dos diferentes aspectos de uma mesma entidade real, representados em diferentes bancos de dados, sem que as modificações locais requeridas em metodologias de integração de banco de dados heterogêneos se fizessem necessárias.

A comparação entre os diferentes BD envolvidos levou à identificação das diferentes representações locais de uma mesma entidade do mundo real, como por exemplo, diferentes informações administrativas de um paciente. Nessas representações equivalentes, adotadas nos diferentes bancos de dados, podem existir semelhanças e conflitos de representação. Nesse contexto, a resolução de conflitos e a unificação de esquemas num esquema global, adotadas em metodologias de integração, foram substituídas pela criação de funções de correlação entre diferentes estruturas de esquemas diferentes. O relativismo semântico permitiu diferentes interpretações da realidade. As modificações locais de esquemas não foram consideradas para efeito de mapeamento, de modo que autonomia local não foi ameaçada.

Projeto Conceitual do Banco de Dados

O modelo do hospital possui cinco entidades: hospitais, médicos, pacientes, convênios e tipo de serviços. As razões de cardinalidade para o modelo entidade relacionamento da Figura 20 são todos N:M.

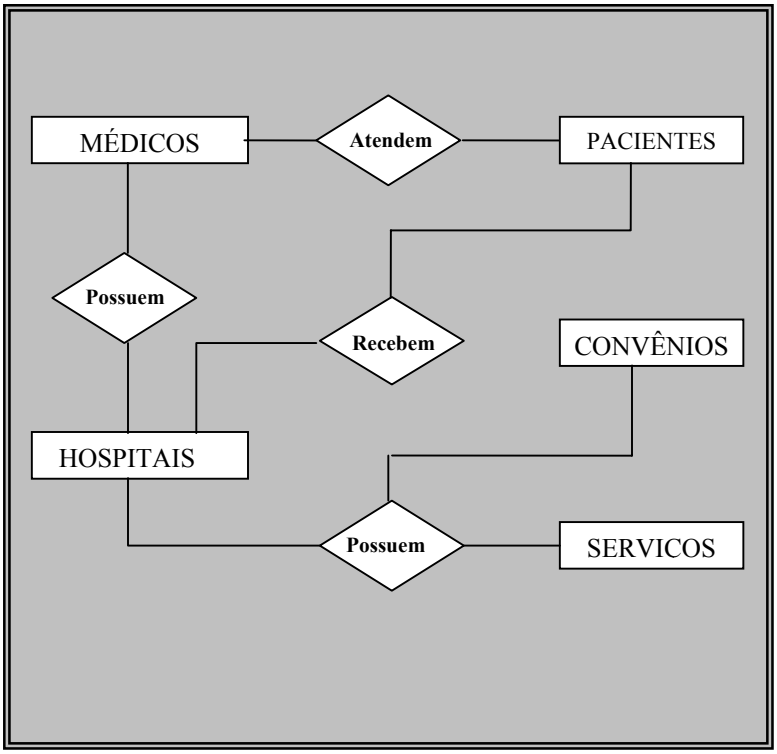



**Figura 20 –** Modelo Entidade - Relacionamento

As Tabelas de 1 a 4 ilustram o projeto conceitual de cada uma das entidades existentes no banco de dados.

.

| | |
|---|---|
| HOSPITAIS | É um tipo de entidade que possui os atributos Nome_Hosp, TipoServ_Hosp, Cod_Hosp, End_Hosp, Tel_Hosp, Serv_Hosp, Conv_Hosp e Leito_Hosp. End_Hosp, Serv_Hosp, Conv_Hosp e LeitoHosp são atributos compostos. Além disso, pode-se especificar que Cód_Hosp é atributo chave, porque foi especificado para ser único. |

**Tabela 1 –** Projeto Conceitual da Entidade Hospitais

| | |
|---|---|
| SERVIÇOS | É um tipo de entidade que possui os atributos Nome_Serv, Cod_ Serv. Pode-se especificar que Cód_ Serv é atributo chave, porque foi especificado para ser único. |

**Tabela 2 –** Projeto Conceitual da Entidade Serviços

| | |
|---|---|
| CONVENIOS | É um tipo de entidade que possui os atributos Cod_ Conv e Nome_ Conv. Pode-se especificar que Cód_ Conv é atributo chave, porque foi especificado para ser único. |

**Tabela 3–** Projeto Conceitual da Entidade Convênios

| | |
|---|---|
| PACIENTES | É um tipo de entidade que possui os atributos Cod_ Pac, Nome_ Pac, Tel_ Pac, End_ Pac, Sex_ Pac, TSangue_ Pac, DataNasc_Pac, Conv_ Pac e NomeResponsavel_Pac. Pode-se especificar que Cód_ Pac é atributo chave porque foi especificado para ser único. E que os atributos Tel_ Pac, End_ Pac são atributos compostos. Além disso, TSangue_ Pac é um atributo nulo desconhecido e Tel_Pac é um atributo nulo não aplicável, pois não é sabido se o seu valor existe. |

**Tabela 4–** Projeto Conceitual da Entidade Pacientes

Criação e manutenção do BDD

Para a criação e manutenção do BDD utilizou-se a linguagem PL/SQL, Oracle9*i* do SGBD. O Anexo E mostra o código SQL gerado na criação do BDD Oracle9*i*.

## 5.1. Desenvolvimento e Testes

A linguagem de programação usada para especificar conteúdos a serem lidos por aparelhos sem fio como, por exemplo, telefones celulares foi a linguagem de marcação WML. Apesar de suas limitações, a linguagem possui uma grande variedade de características, incluindo: suporte a texto e a imagem; suporte a entrada de dados do usuário; pilha de história e navegação; suporte internacional (ou seja, seu conjunto de caracteres é o conjunto de caracteres Universais do ISO / IEC ); otimização para banda estreita (isto é, a capacidade de especificar múltiplas interações com o usuário (*cards*) em uma única transparência da rede (*deck*)); gerenciamento de estado e de contexto[WF02]. O Anexo C contém o código WML gerado.

No caso da aplicação WML desenvolvida neste projeto, o usuário navegará em um conjunto de *cards* WML, os quais poderão conter instruções que poderão invocar serviços no servidor. Os *decks* WML poderão ser requisitados ao servidor conforme sejam necessários, além disso, poderão ser armazenados em arquivos estáticos ou poderão ser gerados dinamicamente por

geradores de conteúdo, como por exemplo, *scripts* ou *servlets*[6]. A Figura 21 ilustra os contextos nos quais o usuário poderá navegar.

[6] *Servlets são classes Java que obedecem a uma certa interface (implementam alguns métodos)* e são invocados por algum *servelet engine, configurado para encontr-la a*

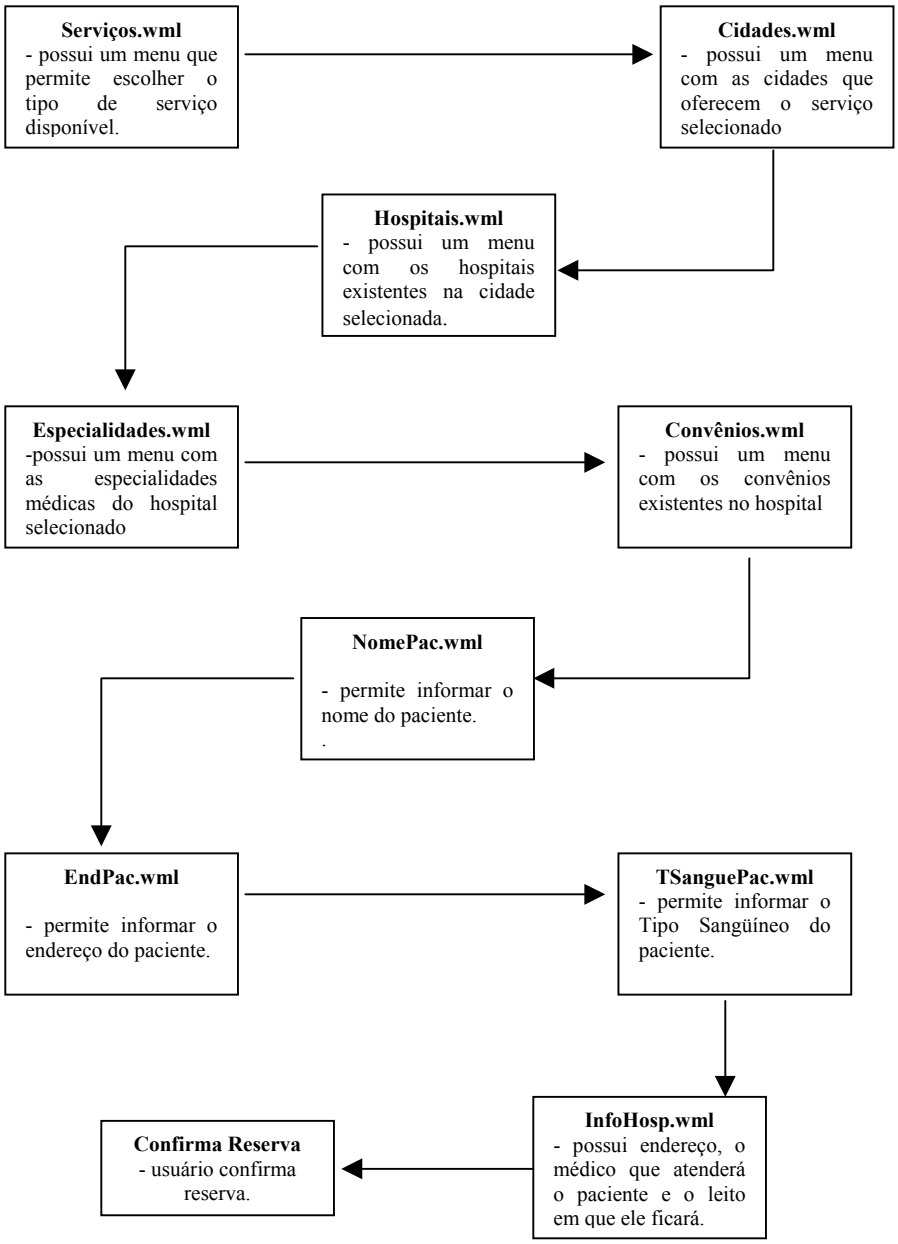



**Figura 21:** Projeto de Navegação da Aplicação

---

*partir de uma URL.*

Editores

Para o desenvolvimento da parte estática da aplicação WML, foi preciso usar um editor de código WML. E, dentre os editores existentes, o *Easy Pad Waptor* [Wtor02] foi o escolhido para ser usado. A justificativa para tal escolha baseou-se no fato de diversas fontes, o classificarem como sendo o mais completo editor WML [Cas00].

O *Easy Pad Waptor* possui uma interface amigável e simples, permite a pré-visualização dos resultados e, além disso, é gratuito. Apesar destas características terem sido comprovadas e, acima de tudo, serem respeitadas, o *Easy Pad Waptor* não oferece uma confiabilidade satisfatória na correção de erros. Ou seja, a pré-visualização dos resultados muitas vezes não retrata fielmente, o conteúdo fonte do código. Uma forma encontrada para contornar este problema foi o uso de outro editor WML, o *DreamWeaver MX* da Macromedia. Este editor resolveu o problema de identificação de erros no código WML. Assim, após a constatação da não existência de erros de sintaxe no código WML, através do *DreamWeaver MX,* o código era transferido para o *Easy Pad Waptor* para uma pré-visualização correta dos resultados.

O Anexo A mostra trechos do código WML vistos nos dois editores e o Anexo B apresenta a pré-visualização de telas feitas em diversos tipos de dispositivos.

Simuladores

São *softwares* que simulam o funcionamento de dispositivos móveis. São usados na fase de testes da aplicação. Portanto, dentre os vários simuladores disponíveis, o SPARH foi testado no WINWAP , o qual permitiu a visualização da aplicação, como seria no telefone celular. O WINWAP se comporta como um *browser*, semelhante ao Netscape Communicator, porém apenas para páginas WML. A cópia deste simulador usada neste projeto, foi adquirida em [Win02].

A plataforma de desenvolvimento Oracle9*i*

De acordo com o que foi descrito no Capítulo 4, não há como contestar que o Oracle9*i* é uma plataforma adequada no desenvolvimento de aplicações *wireless* por conter um módulo específico para tal propósito evitando o ônus de ter que utilizar outra ferramenta.

# CAPÍTULO 6 CONCLUSÕES

A proposta inicial do trabalho foi parcialmente concluída, restando apenas implementar a comunicação entre o BD Oracle e o dispositivo móvel, porque os problemas com as ferramentas de desenvolvimento impossibilitaram tal conclusão. No entanto, o BDD dos hospitais foi modelado e implementado no Oracle9*i*. Toda a arquitetura do protocolo WAP foi detalhada e relacionada com a funcionalidade da arquitetura cliente-servidor dos BDD. Foi um período em que as maiores dificuldades encontradas, indiscutivelmente, relacionaram-se à parte técnica, referentes às instalações de sistemas operacionais e extensos *downloads*. Tudo isso despendeu muito tempo e trabalho.

O primeiro obstáculo foi fazer os *downloads* [Ora02] dos discos de instalação do Oracle9*i*. O volume muito grande dos arquivos de instalação do Oracle9*iDatabase* e do Oracle9*i AS* exigiram muito tempo de espera. A Tabela 5, contém o tamanho exato dos discos de instalação das versões Oracle9*iDatabase* e do Oracle9*i AS*. O fato da máquina em uso suportar somente a versão para *Windows* NT, fez com que o desperdício de tempo com formatações e instalações de sistemas operacionais fosse enorme. Concluída esta primeira, grande e problemática etapa partiu-se, então, para a fase de instalação da ferramenta Oracle.

## Oracle9*i* Application Server Enterprise Edition for Win NT/2000

- **Oracle9*i* Application Server Release 2 for Windows NT/2000** *(1,485,963,785 bytes)*

## OR, Download the file segments

**Disk 1**
**Disk 2**
**Disk 3**

- disk1.zip.001 *(188,743,680 bytes)*
- disk1.zip.002 *(188,743,680 bytes)*
- disk1.zip.003 *(161,547,148 bytes)*
- disk1.zip.BAT *(280 bytes)*

- disk2.zip.001 *(188,743,680 bytes)*
- disk2.zip.002 *(188,743,680 bytes)*
- disk2.zip.003 *(110,830,646 bytes)*
- disk2.zip.BAT *(280 bytes)*

- disk3.zip.001 *(188,743,680 bytes)*
- disk3.zip.002 *(188,743,680 bytes)*
- disk3.zip.003 *(81,043,507 bytes)*
- disk3.zip.BAT *(280 bytes)*

## Oracle9*i* Database Release 2 Enterprise/Standard/Personal Edition for Windows NT/2000/XP

**Download the Complete Files**

92010NT_Disk1.zip (612,802,971 bytes)

92010NT_Disk2.zip (537,604,934 bytes)

92010NT_Disk3.zip (254,458,106 bytes)

**Tabela 5** – Descrição do tamanho dos discos de instalação Oracle9*i*

No momento em que iniciou-se a instalação da ferramenta Oracle9*i* viu-se que as versões do Oracle9*iDatabase* e do Oracle9*iAS* eram incompatíveis. A Figura 22 mostra o ponto exato em que o erro aconteceu. E, diante dessa situação, buscou-se soluções juntamente à Oracle, mas infelizmente, não houve retorno e o tempo de desenvolvimento chegara ao fim. Assim, por ter encontrado inúmeros problemas técnicos, não foi possível atingir a meta de desenvolver a aplicação, fazendo com que a extensão pretendida para o projeto não fosse atingida. Entretanto, fica como sugestão para trabalhos futuros o desenvolvimento desta aplicação que aqui não pôde ser concluída.

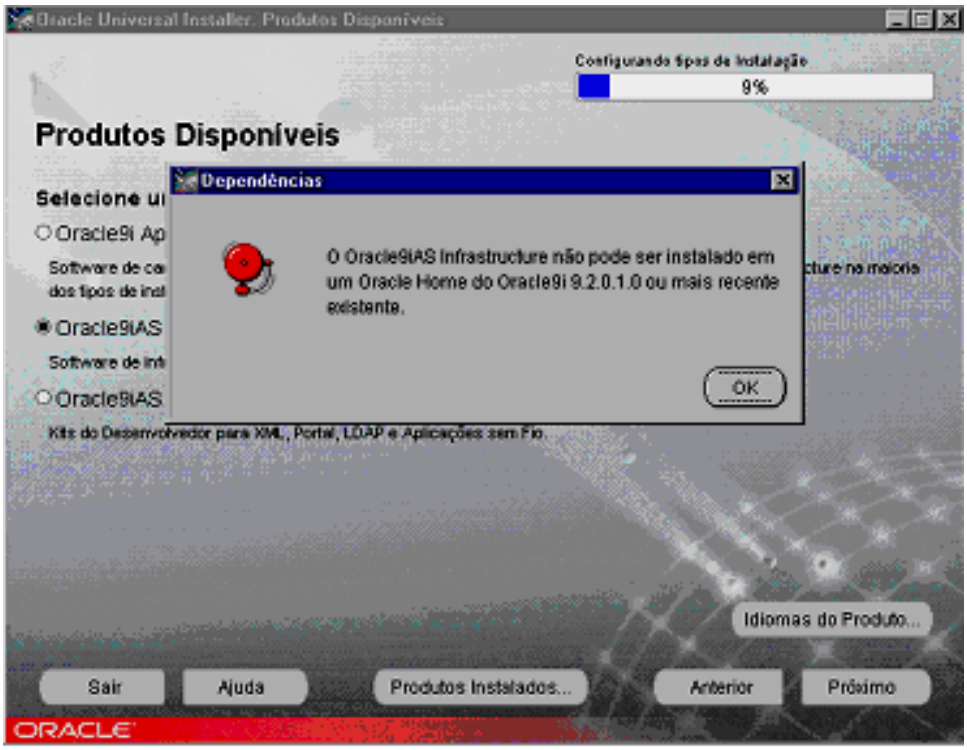



**Figura 22 –** Erro ocorrido na instalação do Oracle9*i*AS

Embora não tenha sido possível instalar o Oracle9*iAS* que contém o componente *wireless* todo o banco de dados da aplicação foi desenvolvido no componente Oracle9*iDatabase.* O Anexo D mostra as telas da que comprovam a construção do banco de dados no Oracle9*i Enterprise Manager Console*.

Não fosse o curto intervalo de tempo para a conclusão deste projeto, esta atual versão poderia ser melhorada em vários pontos. Por exemplo, muitos outros novos atributos poderiam ser inseridos ao banco de dados, novas

consultas poderiam ser realizadas, outras interfaces para dispositivos poderiam ser desenvolvidas, dentre outros.

Baseando-se e aperfeiçoando-se este trabalho, outras aplicações semelhantes e interessantes também poderão ser iniciadas. Como exemplo, temos: **sistema para a polícia**, a facilidade de acesso à ficha criminal do bandido no ato da apreensão; **sistema para vendedores ambulantes**, a confirmação instantânea da venda; **sistema para pesquisadores do censo** atualização imediata das informações sobre determinado local etc. São muitas as possíveis implementações, pois se trata de uma tecnologia cômoda e conveniente a todos.

## CAPÍTULO 7 REFERÊNCIA BIBLIOGRÁFICA

[Cer85] CERI, S., ***Distributed Database - Principles and Systems***. McGraw-Hill, 1985.635p.

[CD91] COULOURIS, G. F.; DOLLIMORE, J., ***Distributed Systems - Concepts and Design***. International Computer Science Series, Addison-Wesley, 1991.308p.

[Spo02] SPOSITO, G., ***Wireless*BR – Desenvolvimento de Aplicações WAP**. Brasil, Agosto de 2000. Disponível em **http://sites.uol.com.br/helyr/sposito**. Pesquisado em: Agosto 2002.

[EN02] ELMASRI, R.; NAVATHE,B.S – **Sistemas de Banco de Dados – Fundamentos e Aplicações**. 3.ed, Rio de Janeiro: LTC Livros Técnicos e Científicos Editora S.A, 2002. 837p.

[WF02] WAPFORUM, ***Wireless Protocol Specification***.USA,2000.Disponível em **http://www.wapforum.com.** Pesquisado em: Abril 2002.

[Ora02] ORACLE, **Oracle9i*AS Wireless:Creating a Mobilized Business***.USA, March 2002.Disponível em **http://www.oracle.com** Pesquisado em: Out 2002.

[Cas00] CASTELLANI, M.F. **Linguagem WML**. Brasil,2000. Disponível em **http://www.wapplande.eng.br.** Pesquisado em: Set 2002.

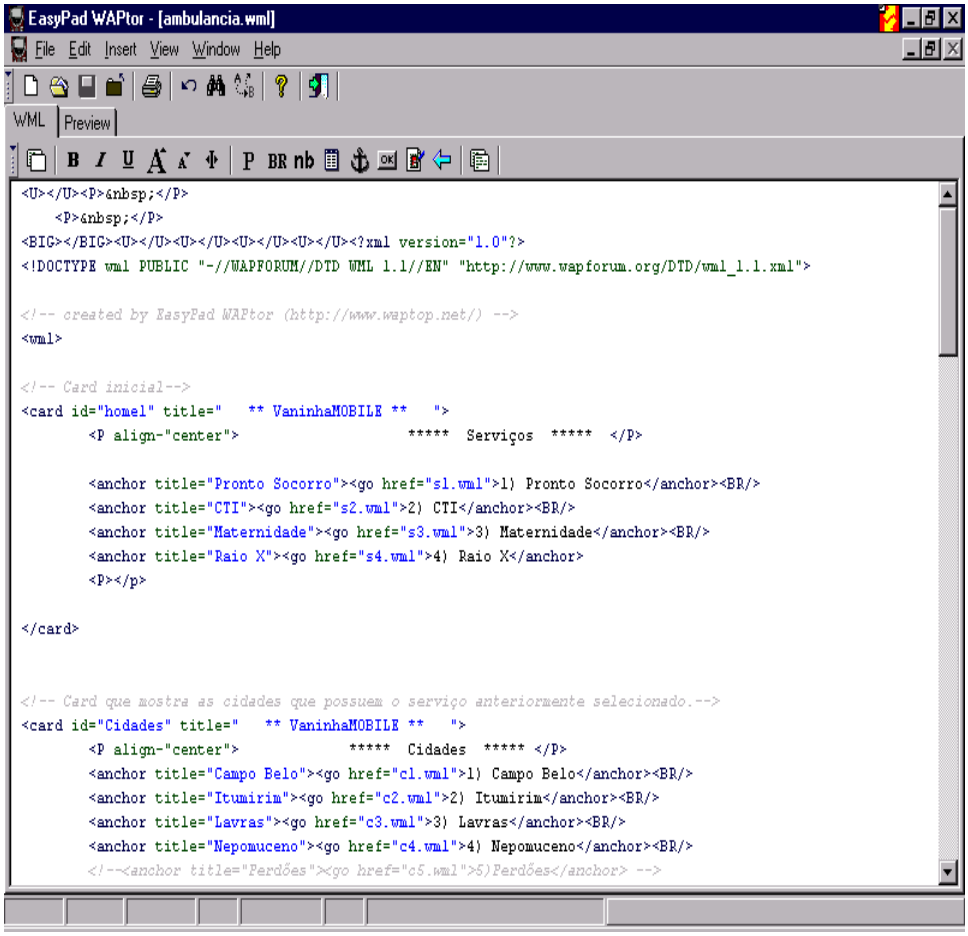
[Win02] WINWAP. **Download do Simulador WINWAP 3.0 PRO.** USA, 2002. Disponível em **http:// www. winwap.org/winwap/features.html.** Pesquisado em: Set 2002.

[Wtor02] WAPTOR. **Download do Editor Easy Pad Waptor.** USA, 2002.

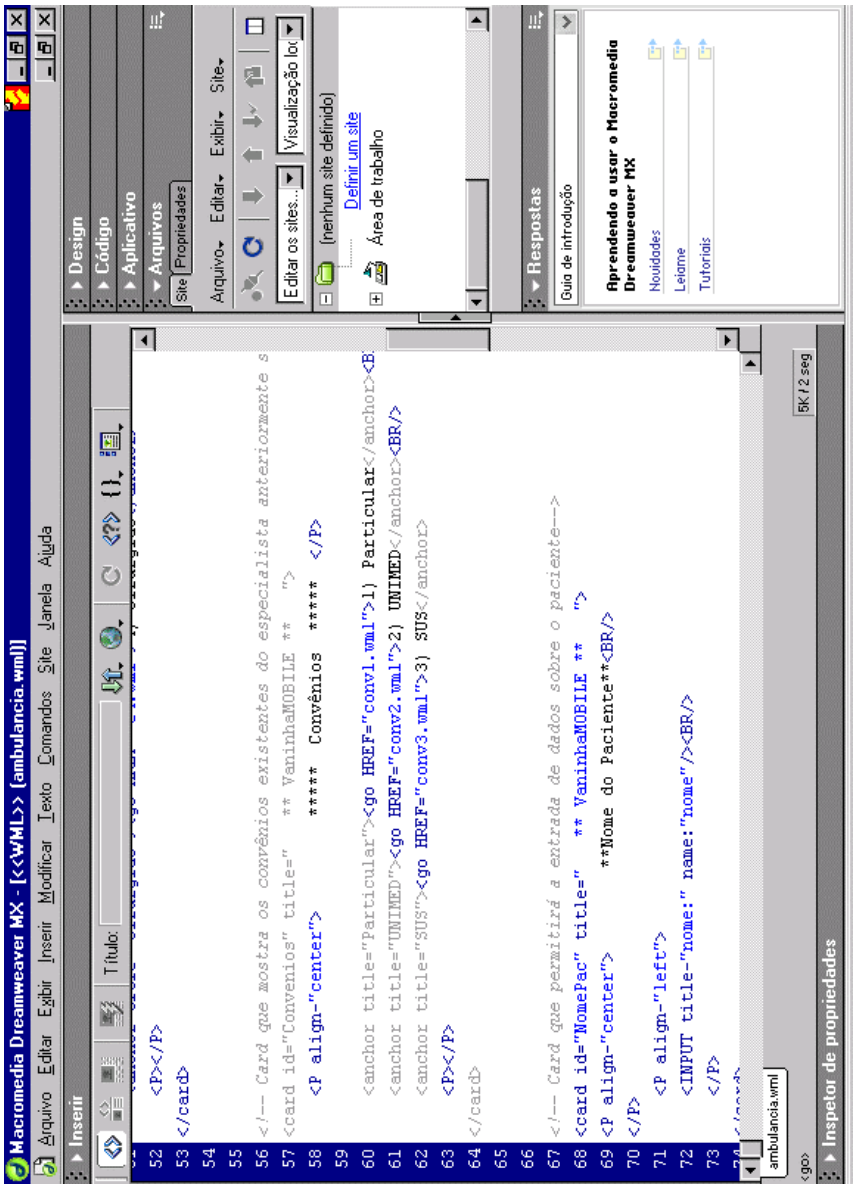
Disponível em **http://www.waptop.net/wap.html**. Pesquisado em: Set 2002.

# ANEXO A

## 1 Editor Easy Pad Waptor

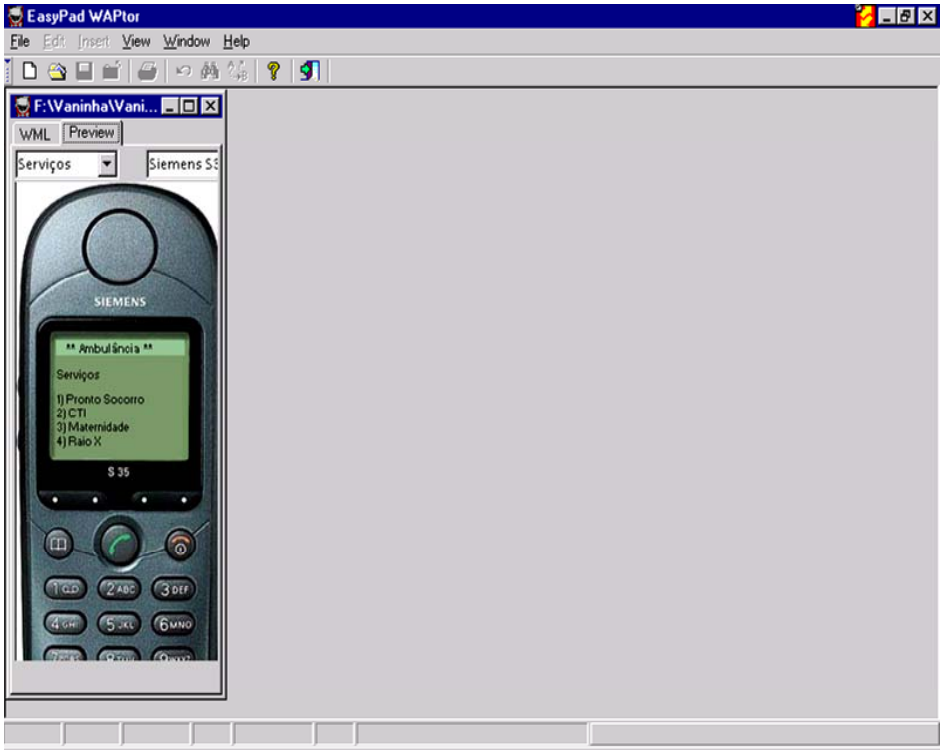



**2. Editor DreamWeaver MX**

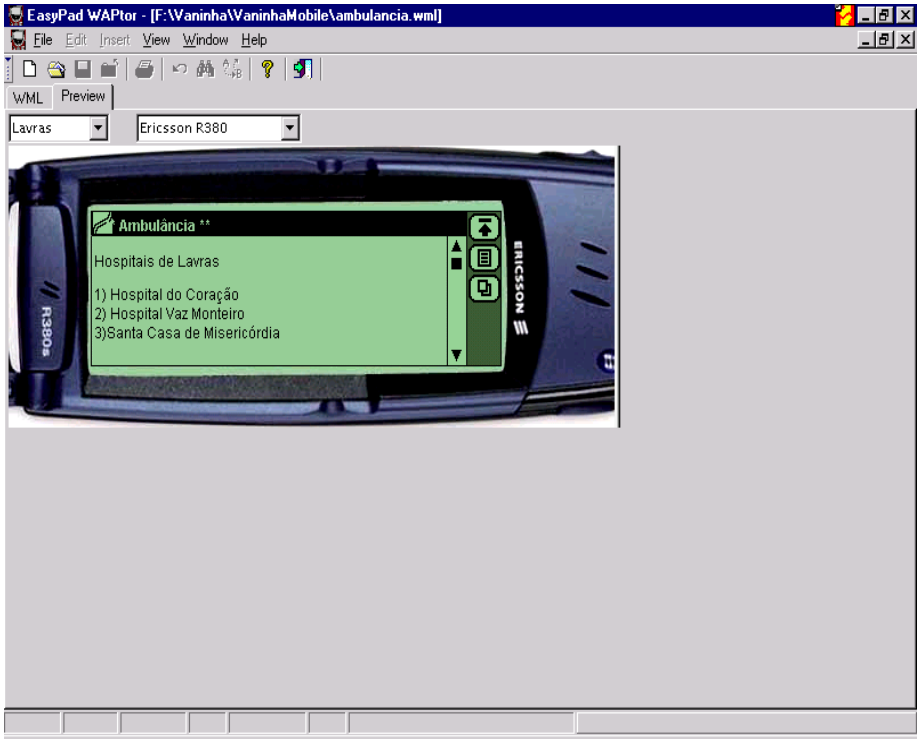

Macromedia Dreamweaver MX - [<<WML>> (ambulancia.wml)]
Arquivo Editar Exibir Inserir Modificar Texto Comandos Site Janela Ajuda
Inserir
Título:
<P></P>
</card>
<!-- Card que mostra os convênios existentes do especialista anteriormente s
<card id="Convenios" title="  ** VaninhaMOBILE **  ">
<P align="center">  *****  Convênios  *****  </P>
<anchor title="Particular"><go HREF="conv1.wml">1) Particular</anchor><B
<anchor title="UNIMED"><go HREF="conv2.wml">2) UNIMED</anchor><BR/>
<anchor title="SUS"><go HREF="conv3.wml">3) SUS</anchor>
<P></P>
</card>
<!-- Card que permitirá a entrada de dados sobre o paciente-->
<card id="NomePac" title="  ** VaninhaMOBILE **  ">
<P align="center">  **Nome do Paciente**<BR/>
</P>
<P align="left">
<INPUT title="nome:" name:"nome"/><BR/>
</P>
ambulancia.wml
<go>
5K / 2 seg
Inspetor de propriedades
Design
Código
Aplicativo
Arquivos
Site
Propriedades
Arquivo Editar Exibir Site
Editar os sites...
Visualização loc
(nenhum site definido)
Definir um site
Área de trabalho
Respostas
Guia de introdução
Aprendendo a usar o Macromedia Dreamweaver MX
Novidades
Leiame
Tutoriais

**ANEXO B**

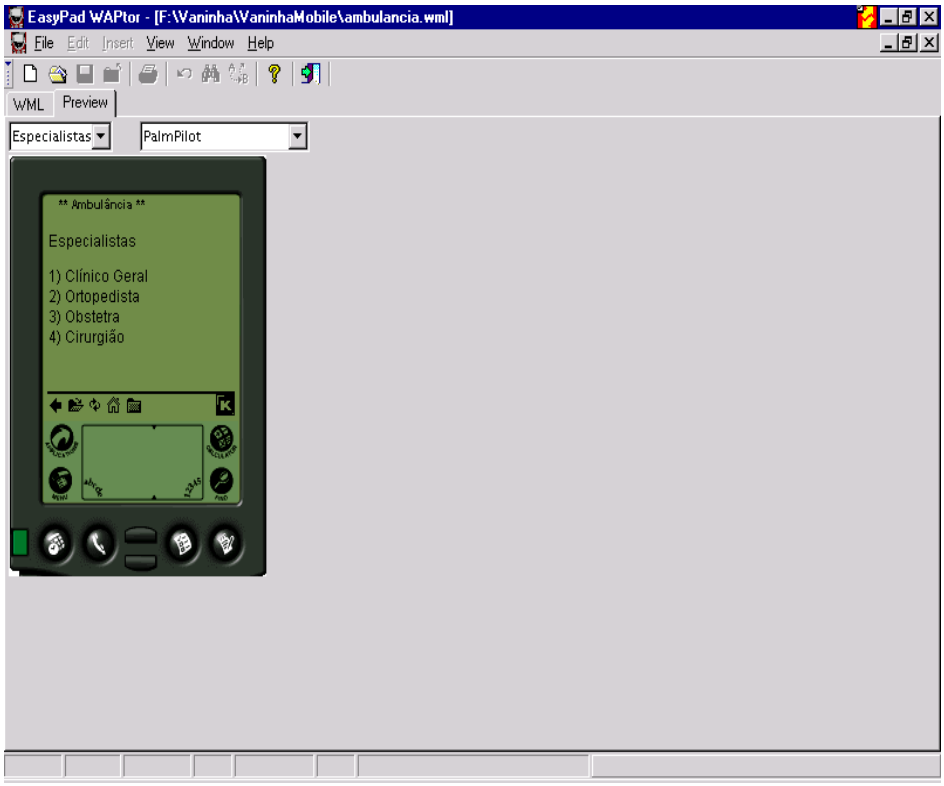
1. **Pré –Visualização do dispositivo Siemens, fornecida pelo editor Easy Pad Waptor.**



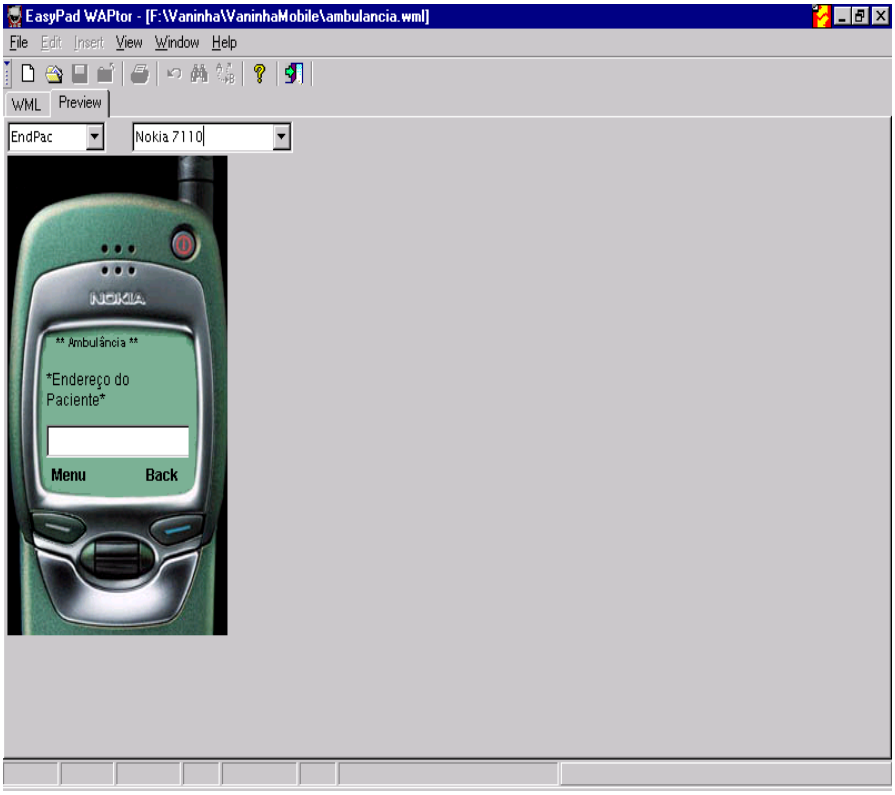

2. **Pré –Visualização do dispositivo Ericsson R380 fornecida pelo editor Easy Pad Waptor.**




**3. Pré –Visualização de um PalmPilot fornecida pelo editor Easy Pad Waptor.**




**4. Pré –Visualização de um Nokia 7110 fornecida pelo editor Easy Pad Waptor.**




## ANEXO C

**1. Código WML correspondente à aplicação da ambulância.**

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<!-- created by EasyPad WAPtor (http://www.waptop.net/) -->


<wml>



<!-- Card inicial que permite escolher o tipo de serviço -->
<card id="Serviços" title="   ** Ambulância **   ">
        <P align-"center"> Serviços</P>

        <anchor title="Pronto Socorro"><go href="s1.wml">
                                1) Pronto Socorro</anchor><BR/>
        <anchor title="CTI"><go href="s2.wml">
                                2) CTI</anchor><BR/>
        <anchor title="Maternidade"><go href="s3.wml">
                                3) Maternidade</anchor><BR/>
        <anchor title="Raio X"><go href="s4.wml">
                                4) Raio X</anchor>
        <P></P>

</card>



<!-- Card que mostra as cidades que possuem o serviço anteriormente
selecionado.-->
<card id="Cidades" title="   ** Ambulância **   ">
        <P align-"center">  Cidades   </P>
        <anchor title="Campo Belo"><go href="c1.wml">
                                1) Campo Belo</anchor><BR/>
        <anchor title="Itumirim"><go href="c2.wml">
                                2) Itumirim</anchor><BR/>

        <anchor title="Lavras"><go href="c3.wml">
                                3) Lavras</anchor><BR/>
        <anchor title="Nepomuceno"><go href="c4.wml">
                                4) Nepomuceno</anchor><BR/>


        <anchor title="Perdões"><go href="c5.wml">
                                5)Perdões</anchor>
        <P></P>
```

```
        </card>


<!-- Card que mostra os hospitais da cidade anteriormente selecionada.-->
<card id="Lavras" title="   ** Ambulância **   ">
        <P align-"center"> Hospitais de Lavras   </P>

        <anchor title="Hospital do Coração"><go href="h1.wml">
                                1) Hospital do Coração</anchor><BR/>
        <anchor title="HospitalVazMonteiro"><go href="h2.wml">
                                2) Hospital Vaz Monteiro</anchor><BR/>
        <anchor title="SantaCasa"><go href="h3.wml">
                                3)Santa Casa de Misericórdia</anchor>
        <P></P>

</card>



<!-- Card que mostra os tipos de especialistas existentes no hospital
anteriormente selecionado.-->
<card id="Especialistas" title="   ** Ambulância **   ">
        <P align-"center"> Especialistas   </P>

        <anchor title="Geral"><go href="e1.wml">
                                1) Clínico Geral</anchor><BR/>
        <anchor title="Ortopedista"><go href="e2.wml">
                                2) Ortopedista</anchor><BR/>
        <anchor title="Obstetra"><go href="e3.wml">
                                3) Obstetra</anchor><BR/>
        <anchor title="Cirurgiao"><go HREF="e4.wml">
                                4) Cirurgião</anchor>
        <P></P>
</card>



<!-- Card que mostra os convênios existentes no hospital selecionado.-->
<card id="Convênios" title="   ** Ambulância **   ">
        <P align-"center">  Convênios    </P>

        <anchor title="Particular"><go HREF="conv1.wml">
                                1) Particular</anchor><BR/>
        <anchor title="UNIMED"><go HREF="conv2.wml">
                                2) UNIMED</anchor><BR/>
        <anchor title="SUS"><go HREF="conv3.wml">
```

```
                                        3) SUS</anchor>
        <P></P>
</card>


<!-- Card que permite informar o nome do paciente-->
<card id="NomePac" title="   ** Ambulância **   ">
<P align-"center"> **Nome do Paciente**<BR/>
</P>
        <P align-"left">
        <INPUT title-"nome:" name:"nome"/><BR/>
        </P>
</card>


<!-- Card que permite informar o endereço do paciente-->
<card id="EndPac" title="   ** Ambulância **   ">
<P align-"center">*Endereço do Paciente*</P>

        <P align-"left">
        <INPUT title="end" name:"end"/><BR/>
        </P>
</card>


<!-- Card que permite informar o tipo sangüíneo do paciente-->
<card id="TSanguePac" title="   ** Ambulância **   ">
        <P align-"left">Tipo Sangüíneo do Paciente</P>

        <P align-"left">
                <INPUT title-"tSangue:"name:"tSangue"/><BR/>
        </P>

</card>
<!-- Card que possui o endereço do hospital, o médico que atenderá e o leito
em que o paciente ficará-->
<card id="Informações" title="   ** Ambulância **   ">
<P align-"center">      *****  Informações  *****
        <P align-"left">
                Endereço do Hospital: <BR />
                Bairro: <BR />
                Médico: <BR />
                Leito: <BR />
```

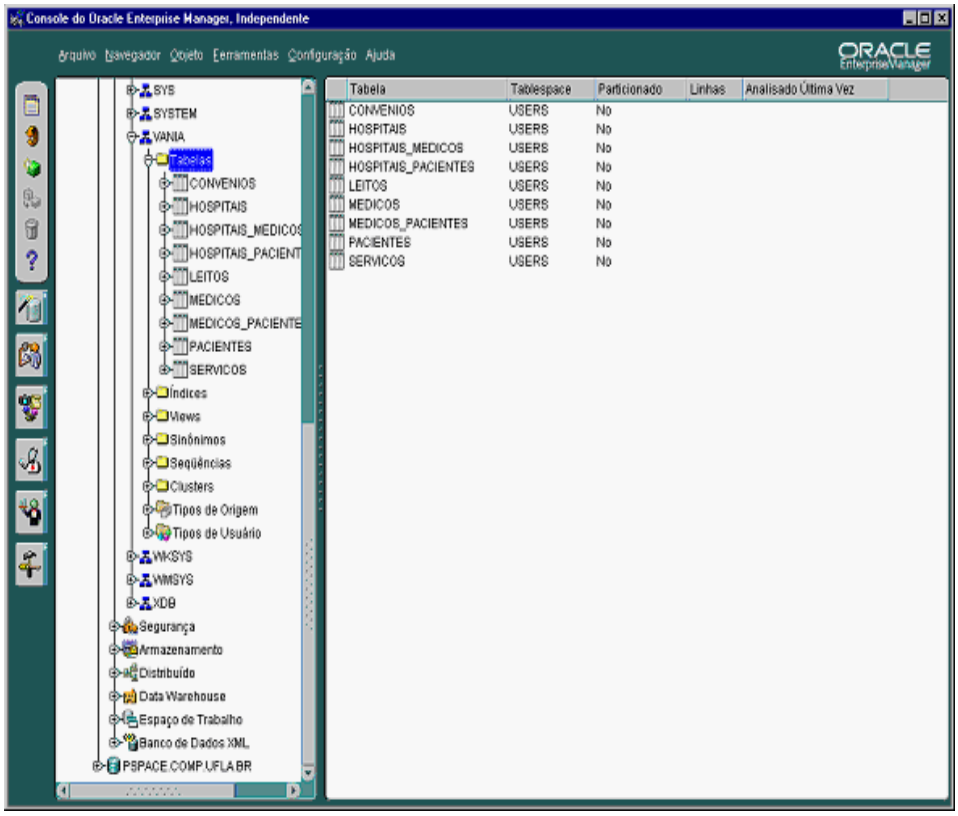
```
        </P>
</card>


<!-- Card que confirma o sucesso da solicitação de reserva solicitada-->
<card id="Confirmação" title="   ** Ambulância **   ">
<P align-"center"> *****  Confirmação *****  </P>

        <P align-"center"> <BR />

                        Reserva feita com sucesso!!!
        </P>

</card>

</wml></go></go></go></go>
```

---

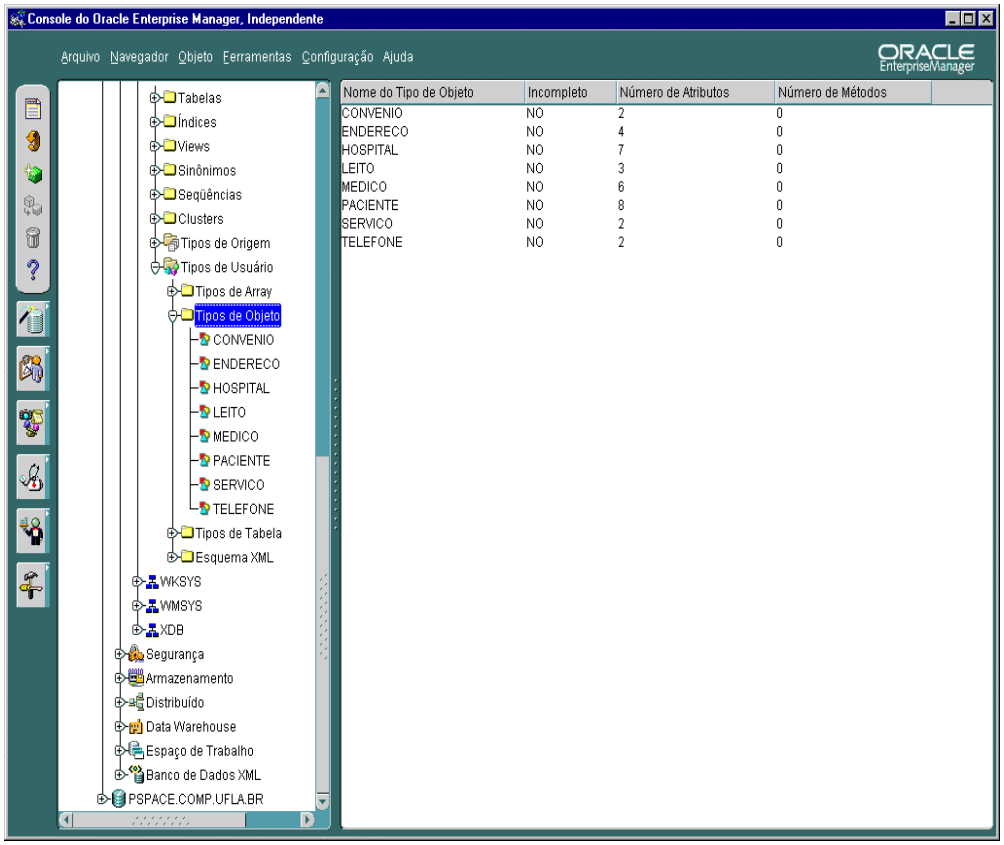
**ANEXO D**

---

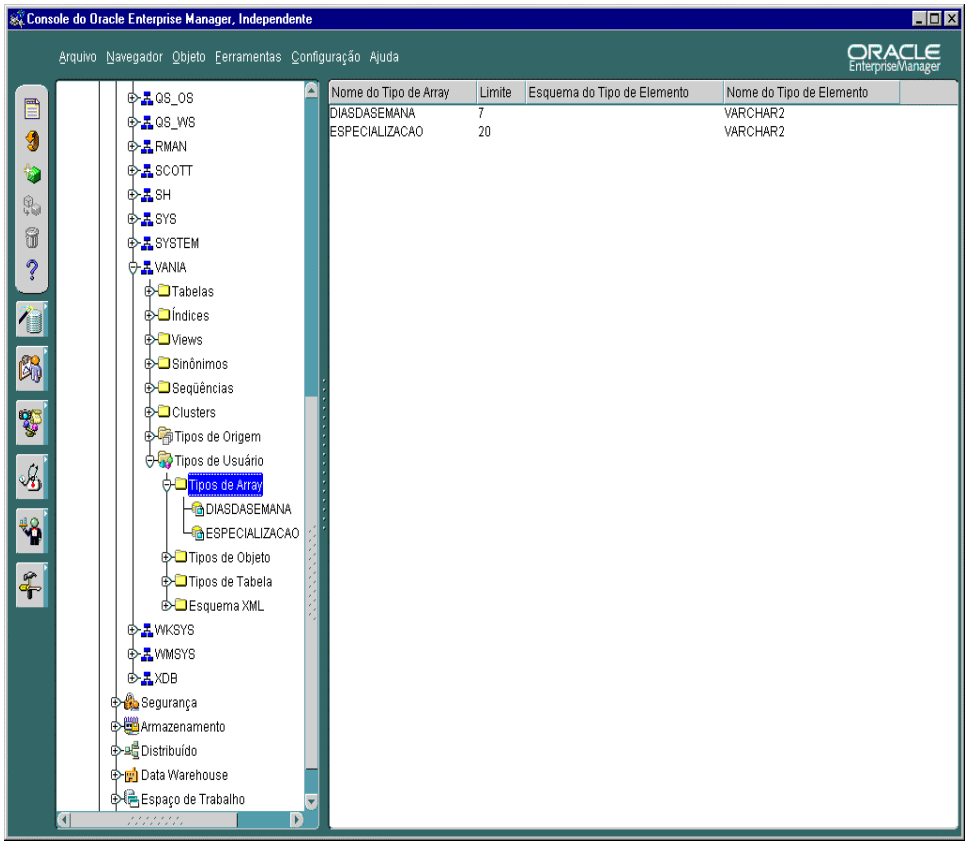
**1. Ilustração das tabelas construídas no Oracle9*i* *Enterprise Manager Console***



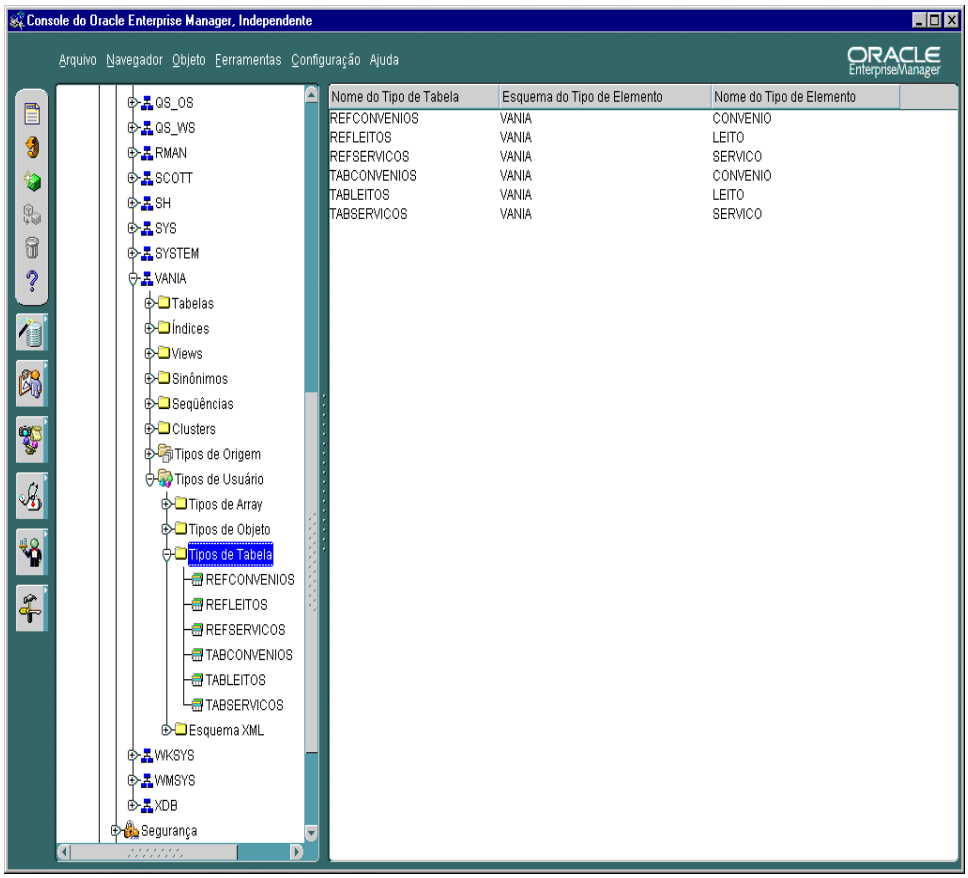

**2. Ilustração dos tipos de objetos construídos no Oracle9*i* *Enterprise Manager Console***



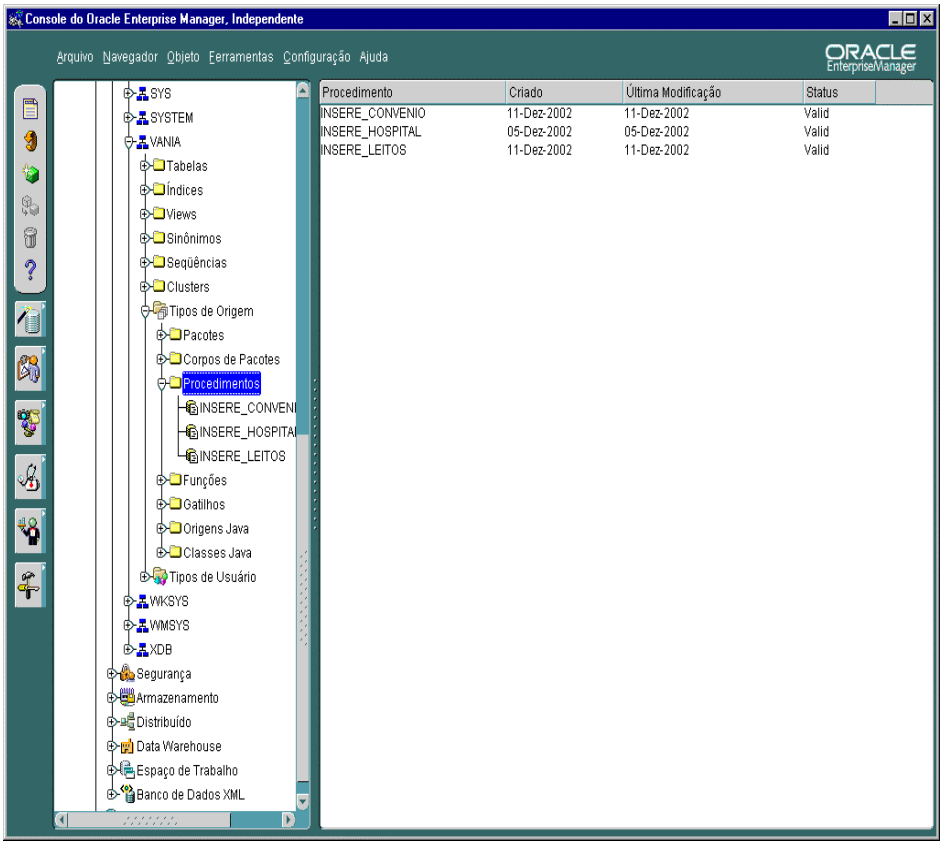

**3. Ilustração dos tipos de array construídos no Oracle9*i Enterprise Manager Console***




**4. Ilustração dos tipos de tabelas construídos no Oracle9*i* *Enterprise Manager Console***




## *5.* Ilustração dos procedimentos construídos no Oracle9*i* *Enterprise Manager Console*


Console do Oracle Enterprise Manager, Independente
Arquivo Navegador Objeto Ferramentas Configuração Ajuda
ORACLE EnterpriseManager
SYS
SYSTEM
VANIA
Tabelas
Índices
Views
Sinônimos
Seqüências
Clusters
Tipos de Origem
Pacotes
Corpos de Pacotes
Procedimentos
INSERE_CONVENI
INSERE_HOSPITA
INSERE_LEITOS
Funções
Gatilhos
Origens Java
Classes Java
Tipos de Usuário
WKSYS
WMSYS
XDB
Segurança
Armazenamento
Distribuído
Data Warehouse
Espaço de Trabalho
Banco de Dados XML
Procedimento
Criado
Última Modificação
Status
INSERE_CONVENIO 11-Dez-2002 11-Dez-2002 Valid
INSERE_HOSPITAL 05-Dez-2002 05-Dez-2002 Valid
INSERE_LEITOS 11-Dez-2002 11-Dez-2002 Valid

## ANEXO E

**Listagem Completa dos comandos Oracle9*i***

**Tipo de Objeto Endereço para ser usado como tipo de dado em outros objetos**

```
CREATE TYPE "VANIA"."ENDERECO" AS OBJECT (
      "RUA"             VARCHAR2(30),
      "NUMERO"          CHAR(6),
      "BAIRRO"          VARCHAR2(20),
      "CIDADE"          VARCHAR2(30))
```

**Tipo de Objeto Tabela de Serviços para ser usado como tipo de dado em outros objetos**

```
CREATE TYPE "VANIA"."SERVICO" AS OBJECT(
      "COD_SERV"        CHAR(5),
      "NOME_SERV"       VARCHAR2(20))
```

**Tipos de tabela de serviços para ser usado em outros objetos, tais como Hospital**

```
CREATE TYPE "VANIA"."REFSERVICOS" AS TABLE OF REF
"VANIA"."SERVICO
```

**Tabela de Serviços**

```
CREATE TABLE "VANIA"."SERVICOS" OF "VANIA"."SERVICO"(
      PRIMARY KEY ("COD_SERV"))
```

**Inserções de Serviços**

```
INSERT INTO SERVICOS VALUES ('00001','PRONTO SOCORRO');
INSERT INTO SERVICOS VALUES ('00002','MATERNIDADE');
INSERT INTO SERVICOS VALUES ('00003','UTI');
INSERT INTO SERVICOS VALUES ('00004','ORTOPEDIA');
```

**Tipo de Objeto Convênio para ser usado como tipo de dado em outros objetos**

```
CREATE TYPE "VANIA"."CONVENIO" AS OBJECT(
      "COD_CONV"        CHAR(5),
      "NOME_CONV"       VARCHAR2(20))
```

**Tipos de tabela de convênios para ser usado em outros objetos, tais como Hospital**

```
CREATE TYPE "VANIA"."REFCONVENIOS" AS TABLE OF REF
"VANIA"."CONVENIO"
```

**Tabela de Convênios**

```
CREATE TABLE "VANIA"."CONVENIOS" OF "VANIA"."CONVENIO"(
      PRIMARY KEY ("COD_CONV"))
```

**Inserções de Convênios**

```
INSERT INTO CONVENIOS VALUES ('00001','SUS');
INSERT INTO CONVENIOS VALUES ('00002','UNIMED');
INSERT INTO CONVENIOS VALUES ('00003','PARTICULAR');
```

**Tipo de Objeto Leito para ser usado como tipo de dado em outros objetos**

```
CREATE TYPE "VANIA"."LEITO" AS OBJECT(
      "COD_LEITO"            CHAR(5),
      "NOME_LEITO"           VARCHAR2(20),
      "NRO_LEITO"            CHAR(3))
```

**Tipo de tabela de leitos para ser usado em outros objetos, tais como Hospital**

```
CREATE TYPE "VANIA"."REFLEITOS" AS TABLE OF REF "VANIA"."LEITO"
```

**Tabela Leitos**

```
CREATE TABLE "VANIA"."LEITOS" OF "VANIA"."LEITO"(
      PRIMARY KEY("COD_LEITO"))
```

**Inserções de Leitos**

```
INSERT INTO LEITOS VALUES ('00001','BERCARIO','001');
INSERT INTO LEITOS VALUES ('00002','BERCARIO','002');
INSERT INTO LEITOS VALUES ('00003','BERCARIO','003');
INSERT INTO LEITOS VALUES ('00004','QUARTO NORMAL','101');
INSERT INTO LEITOS VALUES ('00005','QUARTO NORMAL','102');
INSERT INTO LEITOS VALUES ('00006','QUARTO NORMAL','103');
```

```
INSERT INTO LEITOS VALUES ('00007','APARTAMENTO','301');
INSERT INTO LEITOS VALUES ('00008','APARTAMENTO','302');
INSERT INTO LEITOS VALUES ('00009','APARTAMENTO','303');
```

**Tipo de Objeto Hospital para ser usado como tipo de dado em outros objetos**

```
CREATE TYPE "VANIA"."HOSPITAL" AS OBJECT (
      "COD_HOSP"        CHAR(5),
      "NOME_HOSP"       VARCHAR2(30),
      "TEL_HOSP"        CHAR(15),
      "END_HOSP"        "VANIA"."ENDERECO",
      "SERV_HOSP"       "VANIA"."REFSERVICOS",
      "CONV_HOSP"       "VANIA"."REFCONVENIOS",
      "LEITO_HOSP"      "VANIA"."REFLEITOS")
```

**Tabela Hospitais**

```
CREATE TABLE "VANIA"."HOSPITAIS" OF "VANIA"."HOSPITAL" (
      PRIMARY KEY("COD_HOSP"))
      NESTED TABLE "SERV_HOSP" STORE AS LISTASERVICOS,
      NESTED TABLE "CONV_HOSP" STORE AS LISTACONVENIOS,
      NESTED TABLE "LEITO_HOSP" STORE AS LISTALEITO
```

**Procedimento para Inserir Hospital**

```
CREATE PROCEDURE "VANIA"."INSERE_HOSPITAL" (codigo char,
nome varchar2, telefone char, rua varchar2, numero char,
bairro varchar2, cidade varchar2)
as

begin
  declare
    refservico ref servico;
    listservicos refservicos;
begin
    select ref(sx) into refservico
    from servicos sx
    where cod_serv='00001';
            listservicos := refservicos(refservico);
insert into hospitais (cod_hosp, nome_hosp, tel_hosp,
                       end_hosp, serv_hosp)
      values (codigo, nome, telefone, endereco(rua, numero,
                     bairro, cidade), listservicos);
      commit;
end;
end;
```

**Procedimento para Inserir Convênio**

```
CREATE OR REPLACE  PROCEDURE "VANIA"."INSERE_CONVENIO" (
                  codigo char, nome varchar2)

as
begin
  declare
    refconvenio ref convenio;
    listconvenios refconvenios;

begin
  select ref(sx) into refconvenio
  from convenios sx
  where cod_conv='00001';

  listconvenios := refconvenios(refconvenio);

  insert into convenios (cod_conv, nome_conv)
    values (codigo, nome);
    commit;
end;
```

```
end;
```

**Procedimento para Inserir Leitos**

```
CREATE OR REPLACE  PROCEDURE "VANIA"."INSERE_LEITOS" (
            codigo char, nome varchar2, numero char)

as
begin
  declare
    refleito ref leito;
    listleitos refleitos;




begin
  select ref(sx) into refleito
  from leitos sx
  where cod_leito='00001';

  listleitos := refleitos(refleito);
```

```
  insert into leitos (cod_leito, nome_leito, nro_leito)
          values (codigo, nome, numero);
  commit;
end;
end;
```

**Tipo de Objeto Telefone para ser usado no objeto Médico**

```
CREATE TYPE "VANIA"."TELEFONE" AS OBJECT (
      "RESIDENCIA"      CHAR(12),
      "CELULAR"         CHAR(12))
```

**Tipo de Array de Especialização**

```
CREATE TYPE "VANIA"."ESPECIALIZACAO" AS VARRAY (20) OF
VARCHAR2(30)
```

**Tipo de Array de Dias da Semana**

```
CREATE TYPE "VANIA"."DIASDASEMANA" AS VARRAY (7) OF
VARCHAR2(15)
```

**Tipo de Objeto Médico para ser usado na tabela Médicos**

```
CREATE TYPE "VANIA"."MEDICO" AS OBJECT (
      "COD_MED"         CHAR(5),
      "NOME_MED"        VARCHAR2(35),
      "TEL_MED"         "VANIA"."TELEFONE",
      "ESPEC_MED"       "VANIA"."ESPECIALIZACAO",
      "HORAATEND_MED"   NUMBER(2,0),
      "DIAATEND_MED"    "VANIA"."DIASDASEMANA")
```

**Tabela Médicos**

```
CREATE TABLE "VANIA"."MEDICOS" OF "VANIA"."MEDICO" (
      PRIMARY KEY("COD_MED"))
```

**Tipo de Objeto Paciente para ser usado na tabela Pacientes**

```
CREATE TYPE "VANIA"."PACIENTE" AS OBJECT (
      "COD_PAC"              CHAR(5),
      "NOME_PAC"             VARCHAR2(35),
      "END_PAC"              "VANIA"."ENDERECO",
      "TSANGUE_PAC"          CHAR(2),
```

```
      "SEXO_PAC"              CHAR(1),
      "NOMERESPONSAVEL_PAC"   VARCHAR2(35),
      "DATANASC_PAC"          DATE,
      "CONV_PAC"              "VANIA"."REFCONVENIOS")
```

**Tabela Pacientes**

```
CREATE TABLE "VANIA"."PACIENTES" OF "VANIA"."PACIENTE" (
      PRIMARY KEY("COD_PAC"))
      NESTED TABLE "CONV_PAC" STORE AS LISTACONVPAC
```

**Tabela Hospitais Medicos**

```
CREATE TABLE "VANIA"."HOSPITAIS_MEDICOS" (
      "COD_HOSP"        CHAR(5) NOT NULL,
      "COD_MED"         CHAR(5) NOT NULL,
      PRIMARY KEY("COD_HOSP","COD_MED"),

      FOREIGN KEY ("COD_HOSP")REFERENCES

                        "VANIA"."HOSPITAIS"("COD_HOSP"),
      FOREIGN KEY("COD_MED") REFERENCES
```

```
                        "VANIA"."MEDICOS"("COD_MED"))
```

**Tabela Hospitais Pacientes**

```
CREATE TABLE "VANIA"."HOSPITAIS_PACIENTES" (
      "COD_HOSP"        CHAR(5) NOT NULL,
      "COD_PAC"         CHAR(5) NOT NULL,
      PRIMARY KEY("COD_HOSP","COD_PAC"),
      FOREIGN KEY("COD_HOSP")REFERENCES

      "VANIA"."HOSPITAIS"("COD_HOSP"),
      FOREIGN KEY("COD_PAC")REFERENCES

"VANIA"."PACIENTES"("COD_PAC"))
```

**Tabela Medicos_Pacientes**

```
CREATE TABLE "VANIA"."MEDICOS_PACIENTES" (
      "COD_MED"         CHAR(5) NOT NULL,
      "COD_PAC"         CHAR(5) NOT NULL,
      PRIMARY KEY("COD_MED","COD_PAC"),
      FOREIGN KEY("COD_MED")
            REFERENCES "VANIA"."MEDICOS"("COD_MED"),
```

```
FOREIGN KEY("COD_PAC")

      REFERENCES "VANIA"."PACIENTES"("COD_PAC"))
```