FLÁVIA LOSCHI MARQUES DA SILVA

UM SISTEMA DE PRONTO-ATENDIMENTO HOSPITALAR DISTRIBUÍDO USANDO TECNOLOGIA WIRELESS: IMPLEMENTAÇÃO E AVALIAÇÃO

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

Orientadora
Profa. Olinda Nogueira Paes Cardoso

Co-Orientador
Prof. Doutor Ricardo Martins de Abreu Silva

LAVRAS
MINAS GERAIS – BRASIL
2003

FLÁVIA LOSCHI MARQUES DA SILVA

UM SISTEMA DE PRONTO-ATENDIMENTO HOSPITALAR DISTRIBUÍDO USANDO TECNOLOGIA WIRELESS: IMPLEMENTAÇÃO E AVALIAÇÃO

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

Aprovada em 18 de Dezembro de 2003.

_____

Prof. Doutor Ricardo Martins de Abreu Silva
DCC/UFLA
(Co-Orientador)

_____

Profa. Olinda Nogueira Paes Cardoso
DCC/UFLA
(Orientadora)

LAVRAS
MINAS GERAIS – BRASIL

Resumo

O objetivo deste trabalho consiste em projetar e desenvolver uma camada intermediária (*middleware*) capaz de interligar não apenas *desktops*, mas também dispositivos móveis, a uma base de dados do sistema de pronto-atendimento hospitalar. A interação com *desktops* é feita através do navegador para Web *Internet Explorer*, enquanto as interações com os dispositivos móveis ocorre através de três tipos de tecnologias: WAP (*Wireless Aplication Protocol*), i-Mode, J2ME (*Java 2 Micro Edition*). Os resultados alcançados comprovam a viabilidade e portabilidade deste módulo intermediário, inclusive sendo capaz de ser reaproveitado em outros sistemas de informação.

Sumário

Listas de Abreviaturas e Siglas

ANSI	*American National Standards Institute*
BDD	Banco de Dados Distribuídos
cHTML	*compact HyperText Markup Language*
CPU	*Central Processing Unit*
DBA	*Database Administrator*
DDL	*Data Definition Language*
DML	*Data Manipulation Language*
HTML	*HyperText Markup Language*
HTTP	*HyperText Tranfer Protocol*
J2ME	*Java 2 Micro Edition*
MIDP	*Mobile Information Device Profile*
ODBC	*Open Database Connectivity*
PDA	*Personal Digital Assistants*
RDBMS	*Relational Database Management System*
SBDD	Sistema de Banco de Dados Distribuídos
SGBD	Sistema de Gerenciamento de Dados
SGBDD	Sistema de Gerenciamento de Dados Distribuídos
SQL	*Structured Query Language*
URL	*Uniform Resource Locator*
WAP	*Wireless Application Protocol*
WML	*Wireless Markup Language*
XHTML	*Extensible HyperText Markup Language*
XML	*Extensible Markup Language*

Lista de Figuras

1 Introdução

Nas últimas décadas, a Tecnologia da Informação evoluiu consideravelmente, 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 é o Banco de Dados Distribuídos (BDD) [CM02].

O objetivo deste trabalho é conciliar as tecnologias Banco de Dados Distribuídos e redes sem fio, numa aplicação que atenda às necessidades estratégicas, econômicas e sociais de determinada região. Portanto, a intenção deste projeto é integrar sistemas autônomos e heterogêneos via Internet e acessá-los via rede *Wireless*.

Ninguém está livre de sofrer um acidente ou, de repente, no meio da rua, ter um ataque cardíaco ou um acidente vascular cerebral. Quando isso acontece, o normal é chamar uma ambulância para que a vítima seja socorrida com segurança e da forma correta. Porém, ao levar essa vítima para um hospital, as pessoas que estão socorrendo-a podem chegar a esse hospital e se depararem com o sério problema de não haver vagas disponíveis ou médicos preparados para atender o caso em questão.

Desenvolveremos o módulo de interação entre dispositivos móveis e a base de dados do Sistema de Pronto-Atendimento Hospitalar, desenvolvida por Vânia Marçal em 2002 [Mar02]. Este módulo de interação, sugerido pela mesma como trabalhos futuros em sua monografia, consiste em permitir que os socorristas não percam tempo de levar a vítima para hospitais que podem não atendê-la, fazendo uma consulta a um banco de dados, utilizando para isso um

dispositivo móvel, como o celular, por exemplo. Nessa consulta os socorristas ficam sabendo quais hospitais estão aptos a receber tal vítima.

Este módulo de interação torna-se interessante pelo fato de que o sistema de pronto-atendimento hospitalar possui algumas vantagens. Ele é um serviço de utilidade pública, é portável e adapta-se facilmente a outros sistemas de informação. Para isso, é preciso adquirir conhecimentos sobre Banco de Dados e Sistemas *Wireless*.

O módulo de interação é portável tanto quanto aos clientes quanto aos bancos de dados.

- **Portabilidade quanto aos clientes:** eles podem ser o Internet Explorer, i-Mode, J2ME ou WAP.
- **Portabilidade quanto aos bancos de dados:** podemos acessar vários tipos de bancos de dados, como por exemplo, Oracle, Informix, Postgres, Access, MySQL, entre outros.

O módulo se adapta com facilidade a outros sistemas de informação como, por exemplo, sistemas bancários ou qualquer outro sistema que utilize dispositivos móveis.

Essa monografia está dividida da seguinte maneira: no Capítulo 2, é apresentado o Referencial Teórico que trata de Banco de Dados Distribuídos, de Banco de Dados MySQL e da Tecnologia J2ME. No Capítulo 3, temos a Metodologia que aborda a Arquitetura Multi-Tier e a Aplicação Pronto-Atendimento. No Capítulo 4 abordamos os Resultados e Discussões. A Conclusão e os Trabalhos futuros são apresentados no Capítulo 5. E finalmente, no Capítulo 6 temos as Referências Bibliográficas.

2 Referencial Teórico

2.1 Banco de Dados Distribuídos

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.

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.

Defini-se Banco de Dados Distribuídos (BDD) como uma coleção de vários bancos de dados logicamente inter-relacionados, distribuídos ao longo de um sistema de rede de computadores. 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 [Elm02].

2.1.1 Vantagens dos Bancos de Dados Distribuídos

2.1.1.1 Gerência de dados distribuídos com diferentes tipos de transparências

Em termos ideais, um SGBD deve ser transparente na distribuição, de certo modo escondendo os detalhes de 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 os usuários dos detalhes sobre a rede. Pode ser dividida entre Transparência de Localização e Transparência de Nomeação.
- **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:** 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.

2.1.1.2 Confiabilidade e disponibilidade crescentes

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*.

2.1.1.3 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 e de E/S (Entrada/Saída) e ao mesmo tempo reduz a demora no acesso que sistemas de rede de áreas distantes (longo alcance) implicam.

2.1.1.4 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 SBDD como se fosse um único sistema centralizado.

2.1.2 Funções Adicionais de Banco de Dados Distribuídos

Para obter as vantagens potenciais listadas anteriormente, o software do SGBDD deve ter capacidade de fornecer as seguintes funções, alem das funções de um SGBD centralizado:

- Controle dos dados
- Processamento de consultas distribuídas
- Gerenciamento de transações distribuídas
- Gerenciamento de dados replicados
- Recuperação de bancos de dados distribuídos

- Segurança
- Gerenciamento do diretório (catálogo) distribuído

2.1.3 Fragmentação, Replicação e Alocação de Dados

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 distribuídos, criando assim um banco de dados totalmente replicado.

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.

2.1.4 Tipos de Bancos de Dados Distribuídos

Se todos os servidores (ou SGBDs locais individuais) utilizam *softwares* idênticos e todos os usuários (clientes) também 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 de espectro de autonomia, temos um SGBDD que “se parece” com um SGBD centralizado, independente e autônomo que tem seus próprios usuários locais,

transações locais e DBA (*Database Administrator*) e, portanto tem um grau muito elevado de autonomia local.

Ambos os sistemas são híbridos entre sistemas distribuídos e centralizados e a distinção que se faz entre eles não é estritamente seguida.

2.1.5 Visão Geral de Controle de Concorrência e Recuperação em Banco de Dados Distribuídos

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.

2.1.5.1 Controle de concorrência distribuída baseada em cópia distinta de um item de dado

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.

2.1.5.2 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 se próprio bloqueio e pode conceder ou negar o pedido.

2.1.5.3 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 *commit* 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.

2.1.6 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.

Uma série de produtos de SGBD relacional tem adotado uma 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.

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

- 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.
- Cada servidor processa a consulta local e envia a relação resultante para o *site* do cliente.
- O *site* do cliente combina os resultados das subcunsultas para produzir o resultado da consulta originalmente submetida.

Em um SGBD típico, é comum dividir os módulos de *software* em níveis:

- O *software* do servidor
- O *software* do cliente
- O *software* de comunicação

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

2.2 Banco de Dados Oracle

Na arquitetura cliente-servidor, o sistema de banco de dados Oracle é dividido em duas partes:

1. um *front-end* como parte do cliente e;

2. um *back-end* como parte do servidor.

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 isto.

Todos os bancos de dados do Oracle em um sistema de banco de dados distribuídos (SGBDD) utilizam o *software* de rede Net8 para comunicação inter-banco de dados [CD91].

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*** [1] (instantâneos) ou tabelas-mestra replicadas. A replicação é fornecida nos seguintes níveis:

- Replicação básica
- Replicação avançada (simétrica)

2.2.1 Banco de Dados Oracle e Redes Wireless

As características do Oracle9i *Application Server* fornecem a mais flexível, escalável e confiável infra-estrutura móvel do mercado atual, com multi-canais de acesso para qualquer aplicação, incluindo voz. O componente Oracle9iAS *Wireless* existente nessa infra-estrutura é a plataforma de

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

desenvolvimento e disponibilização de aplicações móveis que possui o melhor custo-benefício na utilização da rede *wireless* para comunicação entre terminais móveis e o servidor *Web*.

2.2.1.1 Infra-estrutura *Wireless*

Para implementações simples e de pronta entrega, o Oracle9iAS *Wireless* oferece o componente *Transcoding*, que traduz dinamicamente o conteúdo e aplicações existentes na *Web* em XML e otimiza a aplicação para ser entregue em qualquer dispositivo móvel.

- **Dispositivos Móveis e Browsers**

São usados para acessar a Internet móvel. Todo dispositivo móvel, geralmente usa um *browser* para mostrar informações recebidas.

- **Rede sem Fio**

Rede é a base da infra-estrutura usada pelos transportadores sem fio. Uma característica importante de rede é a largura de banda e o tipo de conexão.

Protocolos são usados para distribuir o conteúdo aos aplicativos.

- **Servidores de Aplicação, Aplicações e Conteúdo**

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

Aplicações / Conteúdo têm uma enorme variedade de formas incluindo informações de bancos de dados, personalização de conteúdos, alertas, e-mail, serviços locais etc.

2.2.1.1.1 O processo de requisição móvel

A Figura 1 ilustra o processo de requisição móvel.

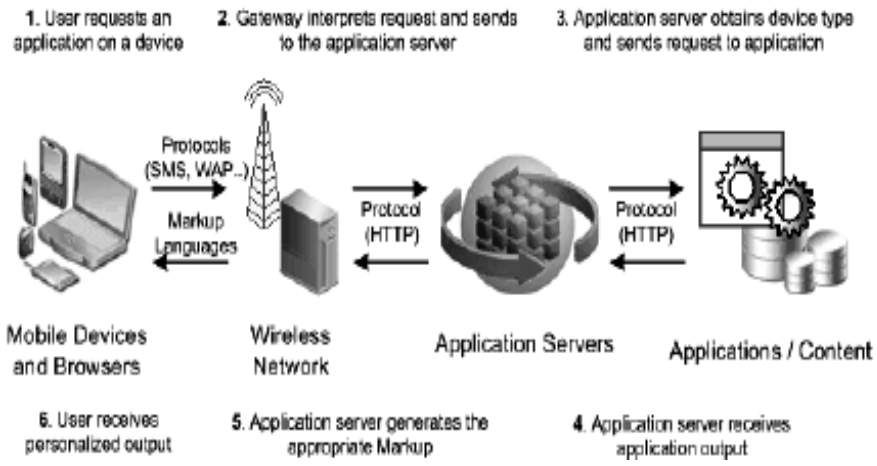

Figura 1 – ***Wireless request process***

As etapas do processo realizado quando um dispositivo móvel solicita um serviço ao servidor de aplicação *wireless* na rede são:

- Envio de requisição
- Reconhecimento e autenticação do dispositivo *Wireless*
- Estabelecimento de uma sessão *Wireless*
- Tradução do pedido para o padrão Internet
- Conexão com o servidor de aplicação
- Reconhecimento da informação sobre o usuário
- Processamento da solicitação *Wireless*
- Customizando o conteúdo para algum usuário

- Adaptação do conteúdo para o dispositivo/rede apropriado

- Localização de Unidades Móveis em Ambientes de Comunicação sem Fio

Existem alguns problemas relativos aos sistemas de comunicação digital sem fio, a saber, aqueles relacionados à localização do usuário. Esses problemas envolvem o planejamento ou definição das Áreas de Localização, as estratégias de atualização da posição do usuário dentro do sistema, e o problema da localização dos usuários em si, o que envolve não só algoritmos específicos, mas também pode envolver um cuidadoso planejamento de banco de dados distribuídos [Roc03].

2.2.1.1.2 Limitações no desenvolvimento de aplicações móveis

Algumas limitações no desenvolvimento de aplicações móveis são citadas abaixo:

- Dificuldade em realizar a entrada de dados em dispositivos móveis
- Tamanho limitado da tela do dispositivo
- Fonte de conteúdos heterogêneos
- Desempenho da aplicação e escalabilidade
- Necessidade de desenvolvimento de um *Wireless Internet Market*
- Evolução de padrões *Wireless*

2.2.2 Oracle9i como solução *Wireless*

O Oracle9iAS faz o *web site* com todas aplicações acessíveis de qualquer *browser* ou dispositivos móveis. O Oracle9iAS *Wireless* é um padrão aberto com soluções integradas para desenvolvimento de aplicações móveis. A Figura 2 mostra a arquitetura do Oracle9iAS *Wireless* [CM02].

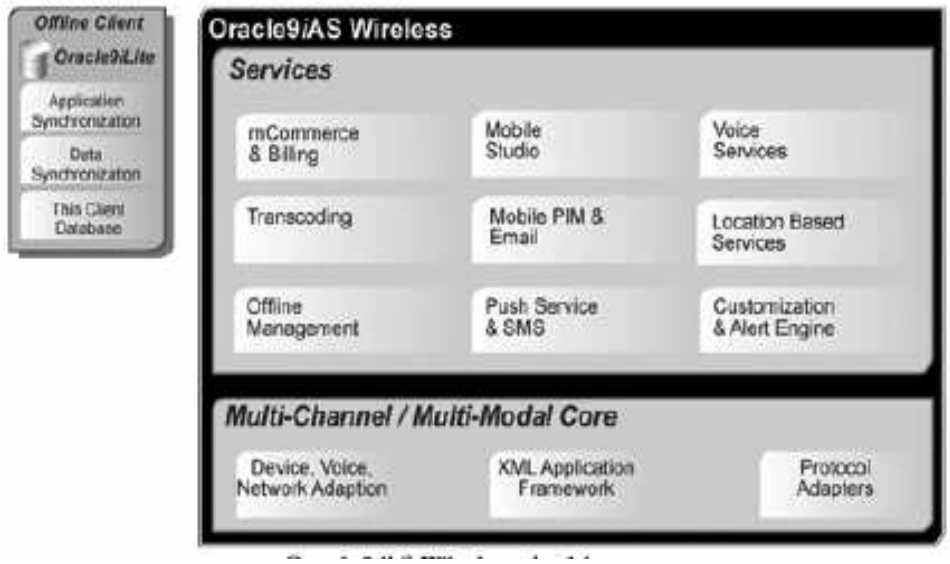

Figura 2 - Oracle9iAS *Wireless Architecture*

2.2.2.1 Desenvolvimento *Multi-Channel* (múltiplos canais), *Multi-Modal* (múltiplas bandas)

Oracle9iAS *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*.

Oracle9iAS *Wireless core* normaliza as complexidades para um protocolo e uma linguagem, o HTTP e o XML.

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.

A ferramenta de aplicação XML multicanal é o segredo para o desenvolvimento único da aplicação para múltiplos canais.

Adaptações em redes e dispositivos transformam e otimizam o conteúdo da aplicação para todo tipo de rede e dispositivo.

Oracle9iAS *Wireless* usa o Oracle9iAS *Database* como repositório para armazenagem de objetos da aplicação. APIs fornecem a funcionalidade de manipular esse armazém de dados.

- 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*.

Os componentes do processo são três: o *gateway* de voz, o servidor de aplicação e o conteúdo fonte. A Figura 3 ilustra o processo de requisição por voz [Mar02].

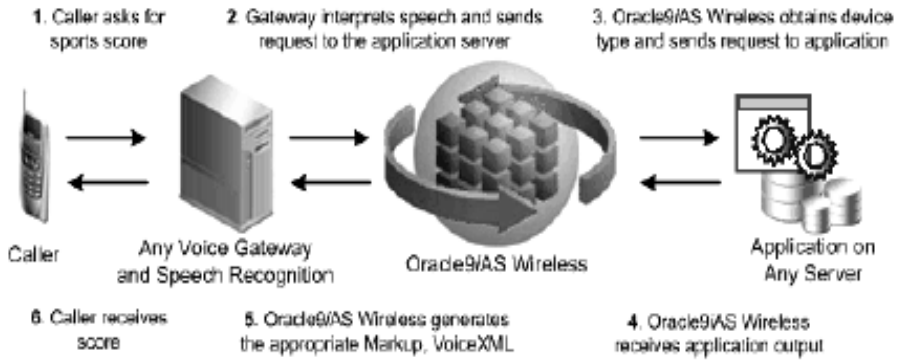

Figura 3 - ***Voice request process***

2.2.2.2 Criação de poderosas aplicações *Push* e SMS

O Oracle9iAS *Wireless 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.

2.2.2.3 Customização da interação entre aplicação e usuário final

O Oracle9iAS *Wireless alert subscription APIs* ou o Oracle9iAS *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.

2.2.2.4 Reuso de aplicações *Web* existentes

Oracle9iAS *Wireless Transcoding 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. O Oracle9iAS *Wireless* traduz o WML em XML, assim torna-se uma linguagem comum para dispositivos móveis que possuem limitações, como padrões de dispositivos.

2.2.2.5 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 restabelecida o usuário pode se conectar ao servidor para atualizar as novas informações. O Oracle9i *Lite* fornece esse tipo de serviço.

2.2.2.6 Noção da localização das aplicações

Aplicações *Location-aware* tomam decisões baseadas na localização geográfica do usuário. Esse serviço é conseguido através do Oracle9iAS *Wireless Location-Based Service*, e não só reduz o número de entradas e baixa o tempo gasto para obter informações, como também melhora a eficiência, possibilitando acesso a informações relevantes para o usuário, tais como mapas, caminhos a seguir, tráfego e serviços.

2.3 Banco de Dados MySQL

O gerenciador de bancos de dados MySQL utiliza a SQL como linguagem de programação.

SQL é a linguagem padrão para acessar sistemas de gerenciamento de bancos de dados relacional (*Relational Database Management System* - RDBMS). A SQL é utilizada para armazenar e recuperar dados para e a partir de um banco de dados.

Há um padrão ANSI para SQL, e o sistema de banco de dados MySQL geralmente se esforça para implementar esse padrão. Há algumas diferenças sutis entre a SQL padrão e a SQL do MySQL. Algumas dessas diferenças são planejadas para se tornarem padrão em versões futuras de MySQL e algumas são deliberadas.

Vez ou outra ouvimos as frases “linguagens de definição de dados” (*data definition language* - DDL), utilizada para definir bancos de dados, e “linguagens de manipulação de dados” (*data manipulation language* - DML), utilizada para consultar bancos de dados. A SQL abrange essas duas bases. A DDL é utilizada ao iniciar a configuração de um banco de dados. Os aspectos de DML da SQL são utilizados muito mais freqüentemente porque essas são as partes que utilizamos para armazenar e recuperar dados reais em um banco de dados [Wel03].

Para instalar o MySql, copiamos o arquivo executável para um diretório do Windows, abrimos o *prompt* do DOS e mudamos para o diretório em questão. Então, é só usar o comando ***mysql --install***.

Para executar o MySql, é só digitar o comando ***mysql.exe***. Depois de ter executado o programa, é preciso criar um banco de dados. Para isso, utiliza-se o comando ***create database <nome_ banco_de_dados>;***.

Uma vez criado o banco de dados, é só conectar a ele, utilizando o comando ***connect <nome_banco_de_dados>;*** e então o banco de dados está pronto para ser utilizado.

A Figura 4 mostra essa seqüência de comandos.

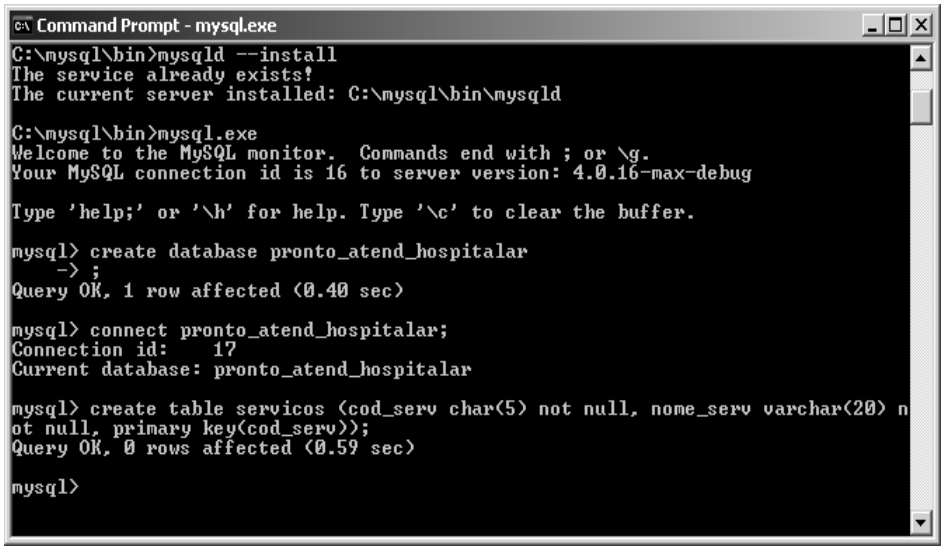

Figura 4 - Instalação e utilização do MySQL

2.4 A Tecnologia J2ME

A plataforma J2ME (*Java 2 Micro Edition*) é utilizada para construir aplicações voltadas para o mercado dos dispositivos com restrições de recursos [Ass03].

Uma implementação em J2ME permite buscar, transferir da Internet e instalar aplicações Java e demais conteúdos.

Com a introdução do Java em dispositivos móveis como telefones celulares e PDAs, temos acesso aos atributos referentes à linguagem e plataforma Java, que é uma linguagem fácil de controlar, um ambiente de

execução que permite uma plataforma segura e portável e acesso a conteúdo dinâmico, além da comunidade de desenvolvedores estimada em dois milhões de pessoas [Gar03] .

2.5 Simuladores e Instruções de Instalação

Para o desenvolvimento de aplicações projetadas para dispositivos móveis, são necessários testes prévios em um computador. Esta sessão discute o uso de simuladores para realizar esses testes. Um simulador mimetiza no computador a aparência, funcionalidade e comportamento de um *microbrowser*.

Vários simuladores gratuitos encontram-se disponíveis na Internet para desenvolvimento e teste de aplicações distribuídas sobre dispositivos móveis. *Openwave* e *Nokia* desenvolveram dois simuladores populares. O simulador *Openwave* é parte do *Openwave Software Developer Kit*, o qual pode ser copiado do seguinte endereço:

http://developer.openwave.com/dvl/tools_and_sdk/openwave_mobile_sdk/index.htm

O *Nokia WAP Toolkit*, contendo o simulador *Nokia*, pode ser copiado do seguinte endereço:

http://forum.nokia.com/main/1,6668,1_1,00.htm

O *Pixo Internet Microbrowser* encontra-se no seguinte endereço:

http://www.pixo.com/products/products001.htm

Para instalar o *Openwave SDK* em uma máquina rodando *Windows*, realize os seguintes passos:

1. Acesse o *site* da *Openwave* e clique em ***Download***.
2. Selecione a opção ***Save this program to disk*** e clique em ***OK***.
3. Forneça um nome e um local para o arquivo e clique em ***Save***.
4. Uma vez que a cópia do arquivo foi completada, clique em ***Open*** para iniciar o processo de instalação.
5. Na tela de boas vindas do programa de instalação, clique em ***Next***.
6. Leia o termo de concordância e clique em ***Yes***.
7. Leia o ***SCREENSHOTS AND IMAGE USE AGREEMENT*** e clique em ***Yes***.
8. Leia o texto na caixa de diálogo de verificação ***Safe Country*** e marque a opção ***Yes*** antes de clicar em ***Next***.
9. Selecione uma pasta de destino na qual será instalado o *Openwave SDK*, clicando no botão ***BROWSE*** ou escolhendo a pasta padrão (esta última é recomendada).
10. Uma pasta para o *SDK* será criada na pasta ***Programs*** do menu ***Start***. Especifique um nome para esta pasta (é recomendado o nome padrão, **UP.SDK 4.1**), e clique em ***Next***.
11. A instalação agora está completa. Veja o arquivo ***README*** e inicie o simulador *Openwave*, cheque cada uma das respectivas opções. Clique em ***Finish***.

Para simular aplicações usando o simulador *Openwave* é necessário um software especializado (chamado servidor *Web*). Os clientes (nesse caso, microbrowser) enviam solicitações de informação do servidor *Web*, e o servidor responde providenciando os recursos (ex., documentos WML, cHTML e documentos HTML). Por exemplo, quando usuários entram com um endereço

URL (*Uniform Resource Locator*) em um microbrowser, eles estão requerendo um documento específico de um servidor *Web*. O servidor *Web* mapeia a URL para um arquivo no servidor (ou para um arquivo na rede de servidores) e retorna o documento requerido para o cliente. Durante esta interação, o servidor *Web* e o cliente se comunicam usando a plataforma independente *HyperText Tranfer Protocol* (HTTP), um protocolo para transferência de requisições e arquivos sobre a Internet.

É importante notar que o simulador *Openwave* requer uma conecção com a Internet. Usuários que não têm acesso a Internet devem considerar o uso do simulador Nokia o qual pode funcionar sem uma conecção com a Internet.

2.6 Arquitetura em Três Camadas

A maior parte das aplicações distribuídas estão baseadas na arquitetura em três camadas, a qual consiste de uma camada cliente (*client-tier*), uma camada intermediária (*business logic* ou *middle-tier*) e uma camada servidor (*server-tier*).

- **Camada Cliente (*Client-Tier*):** freqüentemente criada usando HTML ou HTML Dinâmico. Em alguns casos, esta camada também utiliza Java applets. HTML é o mecanismo preferido para representar a camada cliente em sistemas onde a portabilidade é um problema.
- **Camada Intermediária (*Middle-Tier*):** freqüentemente criada através de servidores Web. Nesta camada está a lógica do sistema que manipula dados do banco de dados e que comunica com o cliente através de navegadores da Web.
- **Camada Servidor (*Server-Tier*):** onde residem as informações, como por exemplo um banco de dados.

Resumindo, através de um navegador da Web, a camada cliente se comunica com a camada intermediária que, por sua vez, pode acessar um banco de dados e manipular seus dados. Todas as três camadas podem residir em computadores distintos conectados por uma rede.

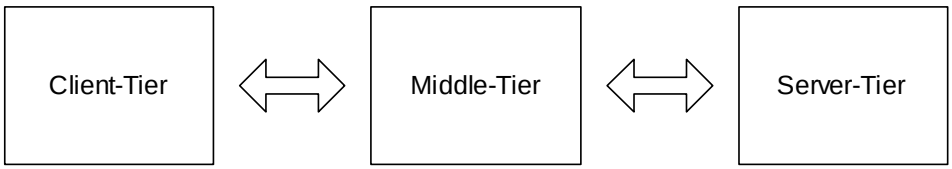

Figura 5 – Arquitetura em três camadas

Em nosso sistema de Pronto-Atendimento Hospitalar Distribuído, cada uma das camadas é criada da seguinte maneira:

- Camada Cliente: pode ser criada usando-se Internet Explorer, i-Mode, WAP e J2ME.
- Camada Intermediária: utilizamos o servidor da Web chamado Apache.
- Camada Servidor: utilizamos o banco de dados MySQL.

3 Metodologia

Este capítulo é baseado no livro *Wireless Internet & Mobile Business – How to Program* [Dei02].

3.1 Aplicação

A aplicação Pronto-Atendimento possui uma arquitetura em três camadas, como mostra a Figura 6.

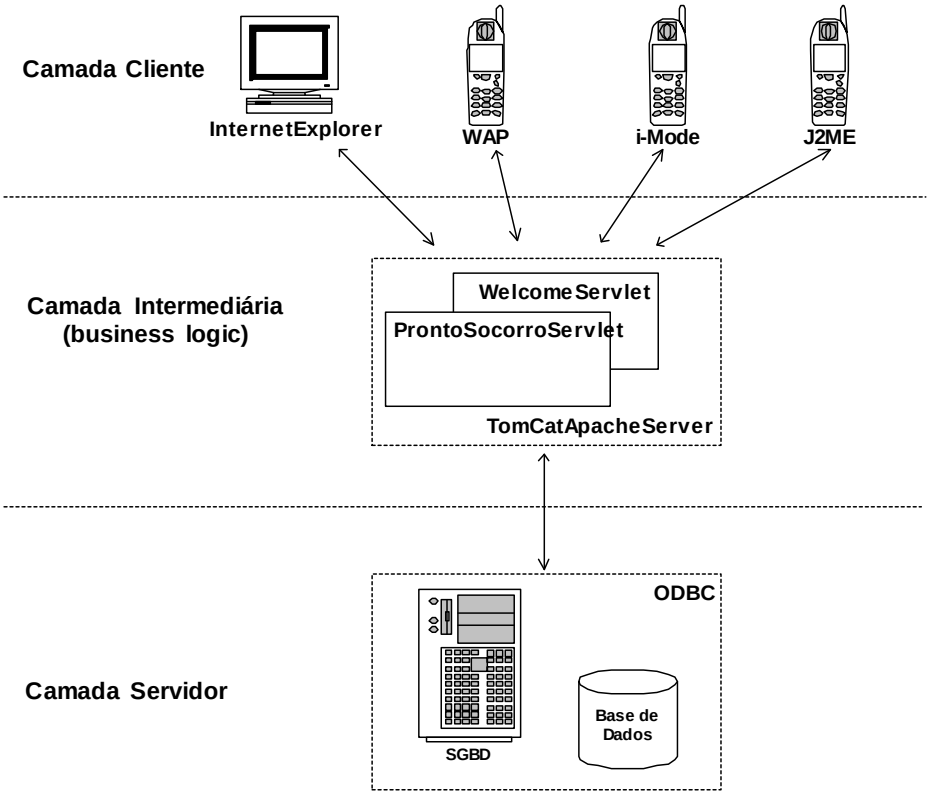

Figura 6 – Arquitetura em três camadas para a aplicação Pronto-Atendimento

A camada cliente consiste de quatro tipos de cliente – Internet Explorer, WAP, i-Mode e J2ME. Cada cliente pode processar um diferente tipo de conteúdo. Microsoft Internet Explorer recebe o conteúdo em XHTML. O simulador Openwave UP é o cliente WAP que recebe o conteúdo em WML. Pixo Internet Microbrowser é o cliente i-Mode que recebe o conteúdo em cHTML. O emulador Sun MIDP-device mostra o cliente J2ME que recebe o conteúdo no formato texto plano. A camada intermediária consiste de dois servlets – **WelcomeServlet** e **ProntoSocorroServlet**. **WelcomeServlet** mostra uma página de boas vindas que introduz a aplicação para o usuário. **WelcomeServlet** também redireciona o cliente para o **ProntoSocorroServlet**.

A camada servidor consiste de um banco de dados que utiliza a consulta que a camada intermediária faz para verificar os dados que precisa enviar de volta e envia a resposta para o **ProntoSocorroServlet**.

3.1.1 Visão Geral do WelcomeServlet

Vamos começar analisando a classe WelcomeServlet, a qual redireciona uma solicitação do cliente para uma tela estática que apresenta as instruções da aplicação Pronto-Socorro – esta tela estática contém um link para o **ProntoSocorroServlet**, o qual permite que o usuário utilize a aplicação.

Os clientes interagem com os servlets fazendo uma série de requisições **get** e **post** para esses servlets. Os clientes enviam requisições **get** para o **WelcomeServlet** e o método **doGet** manipula essas requisições.

Cada cliente recebe uma tela de boas vindas diferente, porque cada tipo de cliente suporta um tipo de conteúdo diferente. Por exemplo, o Internet Explorer recebe um **index.html** como uma tela de boas vindas porque o Internet

Explorer pode processar documentos XHTML. Por outro lado, o simulador Openwave UP recebe um **index.wml** porque um navegador WAP pode processar somente documentos WML. O emulador Sun MIDP-device pode manipular somente textos planos, por isso **WelcomeServlet** envia um **index.txt** para este dispositivo. O navegador Pixo para i-Mode pode processar cHTML, então o servlet envia um **index.html** diferente do que é enviado para o Internet Explorer.

Antes de enviar a tela de boas vindas, o método **doGet** deve determinar qual tipo de cliente fez a requisição. Cada cliente possui um cabeçalho **User-Agent** com cada requisição feita. Esse cabeçalho contém informações dizendo qual tipo de cliente está requisitando dados do servidor.

3.1.2 Configuração do banco de dados

Para registrar o banco de dados MySQL para ser compatível com o ODBC, seguimos os seguintes passos:

1. Abrimos **Start**, **Control Panel**, **Administrative Tools**, **Data Sources (ODBC)** para visualizar a tela **ODBC Data Source Administrator**.
2. Clicamos na pasta **System DNS**, então pressionamos o botão **Add**.
3. Selecionamos na lista o item **MySQL Driver**, então pressionamos o botão **Finish**.
4. Preenchemos o campo **Data Source Name** com "pronto_atend_hospitalar".

4 Conclusões e Trabalhos Futuros

Os resultados alcançados confirmam a viabilidade e portabilidade deste projeto, podendo inclusive ser reaproveitado em outros sistemas de informação. Ou seja, a flexibilidade da utilização do módulo intermediário em diversos bancos dados de dados baseados em ODBC, aliada à sua capacidade de interação com vários tipos de clientes, ressalta a utilidade deste módulo não apenas para o sistema de pronto-atendimento hospitalar, mas em qualquer outro sistema distribuído baseado na arquitetura de três camadas.

As principais dificuldades encontradas dizem respeito: (i) à própria natureza deste trabalho (uma complementação de um trabalho anterior), em que um tempo relevante se faz necessário para análise do antigo projeto; e (ii) os problemas de configuração e suporte relacionados à escolha inicial do SGBD Oracle9iAS para a camada servidora (motivo pelo qual optamos pelo banco de dados MySQL).

Por fim, os trabalhos futuros vislumbrados durante o processo de desenvolvimento referem-se basicamente ao teste de sua adaptação para diversos sistemas de informações baseados em outras tecnologias na camada cliente e servidora.

5 Referências Bibliográficas

[Ass03] ASSIS, W. M., Avaliação da Tecnologia J2ME no Contexto de Desenvolvimento de Jogos *Multiplayers* para Celulares. Departamento de Ciência da Computação – UFLA, 2003, 18p.

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

[CM02] CARDOSO, O. N. P., MARÇAL, V. C., Um Sistema de Pronto-Atendimento Hospitalar Distribuído Usando Oracle9i e WAP. Departamento de Ciência da Computação – UFLA, 2002.

[Dei02] DEITEL, H. M, DEITEL, P. J., NIETO, T. R., STEINBUHLER, K., Wireless Internet & Mobile Business – How to Program. Ed. Prentice Hall, 2002.

[Elm02] ELMASRI, R., NAVATHE, S. B., Sistemas de Banco de Banco de Dados – Fundamentos e Aplicações, Ed. LTC, 3ª. Edição, 2002.

[Gar03] GARROZI, C., Uso da Tecnologia Móvel no Auxílio à Recuperação Alimentar. Departamento de Ciência da Computação – UFLA, 2003, 9p.

[Mar02] MARÇAL,V. C., Proposta de uma Aplicação usando Banco de Dados Distribuídos *Oracle9i* e a Tecnologia WAP das Redes *Wireless*. Departamento de Ciência da Computação – UFLA, 2002.

[Roc03] ROCHA, V. C., Proposta de uma Aplicação usando Banco de Dados Distribuídos Oracle9i e a Tecnologia WAP das Redes *Wireless*. Departamento de Ciência da Computação – UFLA, 2002.

[Wel03] WELLING, L., THOMSON, L., PHP e MySQL – Desenvolvimento Web. Ed. Campus – 2a. Edição, 2003.