VALCIMAR FERREIRA DE CARVALHO

UM ESTUDO COMPARATIVO DAS TECNOLOGIAS *I-MODE* E *WML/WAP* PARA SISTEMAS *WIRELESS* POR MEIO DA IMPLEMENTAÇÃO DE UM SISTEMA DE RESERVAS EM HOTÉIS.

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.

Orientador

Prof. Ricardo Martins de Abreu Silva

LAVRAS
MINAS GERAIS - BRASIL
2002

VALCIMAR FERREIRA DE CARVALHO

UM ESTUDO COMPARATIVO DAS TECNOLOGIAS *I-MODE* E *WML/WAP* PARA SISTEMAS *WIRELESS* POR MEIO DA IMPLEMENTAÇÃO DE UM SISTEMA DE RESERVAS EM HOTÉIS.

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 17 de dezembro de 2002.

_____
Profª. Olinda Nogueira Paes Cardoso
(Co-Orientadora)

_____
Prof. Ricardo Martins de Abreu Silva
(Orientador)

LAVRAS
MINAS GERAIS - BRASIL

Agradecimentos

Agradeço a Deus pela oportunidade e objetivo alcançado, aos meus pais Gonçalo Ferreira de Carvalho e Ana Maria Vitoriana Carvalho, meus irmãos e familiares pelo companheirismo e dedicação nestes quatro anos de curso. Agradeço aos professores Ricardo Martins de Abreu Silva e Olinda Nogueira Paes Cardoso pela amizade e experiência compartilhada na elaboração desta monografia e aos demais professores do DCC e DEX que por bem ou por mal tiveram paciência, disposição e vontade para nos acompanhar durante quatro anos de formação acadêmica.

Quero agradecer também os amigos Fábio e Suênio, atleticanos sofredores, pelo apoio e empréstimo dos computadores e também pelo ambiente fraternal em que vivemos. A minha namorada Polyanna que me acompanhou durante todo o curso, dividindo dificuldades e alegrias. Sempre me incentivando e ajudando a superar barreiras.

Enfim, agradeço a todos meus amigos por essa importante conquista da minha vida.

A todos, o meu muito obrigado.

UM ESTUDO COMPARATIVO DAS TECNOLOGIAS *I-MODE* E *WML/WAP* PARA SISTEMAS *WIRELESS* POR MEIO DA IMPLEMENTAÇÃO DE UM SISTEMA DE RESERVAS EM HOTÉIS.

Resumo

Este trabalho realiza um estudo comparativo entre duas importantes tecnologias *Wireless*, a *WML/WAP* e a *i-Mode*, utilizando como exemplo um Sistema de Reservas em Hotéis com acesso de banco de dados via *Web*. O resultado aqui apresentado além de mostrar as vantagens e desvantagens em se utilizar uma ou outra tecnologia, evidencia as dificuldades para se realizar a migração de um sistema desenvolvido para *desktops* em um sistema para dispositivos móveis.

Palavras Chaves: Simuladores, *Wireless*, *WML/WAP*, *i-Mode*, *cHTML*, *PHP*, *MySQL*.

A COMPARATIVE STUDY Of TECHNOLOGIES *I-MODE* AND *WML/WAP* FOR SYSTEMS *WIRELESS* BY MEANS OF THE IMPLEMENTATION OF A SYSTEM OF RESERVES IN HOTELS.

Abstract

This work carries a comparative study between two important *Wireless* technologies, the *WML/WAP* and *i-Mode*, using as example a System of Reserves in Hotels with access of database through *Web*. The result presented here besides showing the advantages and disadvantages in if using one or another technology, evidences the difficulties to become fullfilled the migration of a system developed for desktops in a system for mobile devices.

Keys Words: Simulators, *Wireless*, *WML/WAP*, *i-Mode*, *cHTML*, *PHP*, *MySQL*.

Sumário

Lista de Figuras

Lista de Abreviaturas e Siglas

AIX	*Advanced Interactive Executive*
API	*Application Programming Interface*
ASP	*Active Server Pages*
BMP	*Bitmap*
CDMA	*Code Division Multiple Access*
CDPD	*Cellular Digital Packet Data*
CGI	*Commum Gateway Interface*
cHTML	*Compact Hyper Text Markup Language*
CSD	*Circuit Switch Data*
CSS	*Cascading Style Sheets*
DTD	*Document Type Definitions*
FOMA	*Freedom of Mobile Multimedia Access*
GSM	*Global System for Mobile Communications*
HP-UX	*Hewlett-Packard Unix*
HDML	*Handheld Devices Markup Language*
HTML	*Hyper Text Markup Language*
HTTP	*Hyper Text Transport Protocol*
IS-136	*Interim Standard - 136*
ITU	*International Telecommunication Union*
JPEG	*Joint Photographic Experts Group*
JSP	*Java Server Page*s
NTT	*Nippon Telegraph and Telephone*
PBM	*Portable Bitmap*
PDAs	*Portable Data Assistent*
PDC-P	*Personal Digital Cellular-Packet*
PHP	*Personal Home Page*
SGBD	*Sistema de Gerenciador de Banco de Dados*
SMS	*Short Message Service*
SSI	*Server-Side Include*s
SSL	*Secure Sockets Layer*

TCP/IP	*Transmission Control Protocol/Internet Protocol*
TDMA	*Time Division Multiple Access*
TLS	*Transport Layer Security*
UDP/IP	*Unreliable Datagram Protocol/Internet Protocol*
URL	*Uniform Resource Locator*
USSD	*Unstructured Supplementary Service Data*
XML	*Extensible Markup Language*
W3C	*World Wide WEb Consortiu*
WAE	*Wireless Application Environment*
WAP	*Wireless Aplication Protocol*
WBMP	*Wireless Bitmap*
W-CDMA	*Wideband Code Division Multiple Access*
WDP	*Wireless Datagram Protocol*
WLAN	*Wireless Local Area Network*
WML	*Wireless Markup Language*
WSP	*Wireless Session Protocol*
WTLS	*Wireless Transport Layer Security*
WTP	*Wireless Transaction Protocol*
WWW	*World Wide Web*

Lista de Tabelas

Capítulo 1

Introdução

Os avanços da comunicação nos últimos anos possibilitaram o surgimento de várias tecnologias, que desde então, procuram atender a real necessidade de seus usuários, com a melhor qualidade possível. No início eram máquinas mono-usuário e muito se teve que evoluir até chegar às redes de computadores atuais. Hoje em dia, as empresas estão apostando numa das mais novas e revolucionárias tendências tecnológicas: a comunicação *Wireless*, as redes sem fio.

A popularidade das redes *Wireless* é crescente. Novas tecnologias têm surgido para atender diferentes necessidades: *Bluetooth*, *HomeRF*, *WAP* (*Wireless Aplication Protocol*), *i-Mode*, *HiperLAN* e *WLAN* (*Wireless Local Area Network*) são alguns exemplos [Mar99]. O mesmo pode-se dizer do surgimento de novos dispositivos tais como: telefones celulares, *pagers*, *PDAs* (*Portable Data Assistant*), entre outros que permitem que seus usuários naveguem por *sites* da Web a qualquer hora e em qualquer lugar.

Uma pesquisa realizada pela *Jupiter Media Metrix* [Jup01], em março de 2001, revelou que o número de usuários de Internet sem fio nos EUA saltaria dos 4,1 milhões, em 2001, para 96 milhões em 2005. De acordo com a pesquisa, em 2005, do total de usuários, 74,9 milhões terão acesso remoto à rede usando telefones celulares, e 7,3 milhões o farão por meio de computadores portáteis. Outros 13,8 milhões navegarão sem fio utilizando organizadores pessoais (*PDAs*). Estendendo para a América Latina, o número de usuários acessando a Internet por telefone celular até 2005, deve ser de aproximadamente 50 milhões, o que tornaria o acesso sem fio tão disseminado na região quanto o acesso baseado em *desktops*.

Porém, o desenvolvimento de aplicações para tais dispositivos é o grande desafio com o qual se deparam os desenvolvedores de *software*. São necessárias diferentes linguagens de marcação (*markup*) tais como: *HTML*

(*Hyper Text Markup Language*) para *PDAs*, *WML* (*Wireless Markup Language*) para a telefones celulares *WAP* e *HTML* compacto (*cHTML*) para telefones *i-Mode*. A escolha da ferramenta correta para o desenvolvimento de uma aplicação tornou-se uma tarefa árdua. Determinar a melhor arquitetura para um sistema atendendo aos objetivos finais demanda tempo e principalmente dinheiro.

O objetivo deste trabalho é apresentar um estudo comparativo entre duas importantes tecnologias *Wireless*, a *WML/WAP* e a *i--Mode*, utilizando como exemplo um Sistema de Reservas em Hotéis com acesso de banco de dados via *Web*. O resultado aqui apresentado, além de mostrar as vantagens e desvantagens em se utilizar uma ou outra tecnologia, evidencia as dificuldades para se realizar a migração de um sistema desenvolvido para *desktops* em um sistema para dispositivos móveis.

O Sistema de Reservas em Hotéis foi escolhido por se tratar de um bom exemplo de aplicação, uma vez que sua principal ferramenta de negócios, o setor de turismo, ocupa um lugar importante no plano governamental de desenvolvimento, devido ao seu impacto econômico e social. Para se ter uma idéia do tamanho da indústria do turismo no mundo, os dados da OMT "Organização Mundial de Turismo" mostram que as viagens internacionais totalizaram em 695,9 milhões, gerando US$ 455,5 bilhões de ingressos de divisas e, segundo a entidade, o setor deve ter crescido em 2001 4% em escala mundial e 3,8% no Brasil. Do ponto de vista da infra-estrutura, o turista brasileiro está sendo bem atendido e, o investimento em novos hotéis, só em São Paulo, alcançará US$ 6 bilhões entre 1998 e 2003. Daí conclui-se que um Sistema de Reservas em Hotéis é um exemplo de aplicação bastante atraente e está completamente dentro das necessidades do mercado [Fig00].

A monografia está dividida em 5 capítulos. O primeiro é uma introdução do trabalho. O segundo capítulo traz uma visão geral das tecnologias *WAP e i-Mode*, apresentando suas perspectivas de mercado e aceitação mundial. Também neste mesmo capítulo, diversas ferramentas que auxiliam na confecção e testes de mini-páginas são citadas sendo algumas delas mais bem detalhadas.

O terceiro capítulo apresenta o modelo desenvolvido para dispositivos móveis, as etapas de implementação, alguns dos comandos SQL mais usados e o servidor de aplicação utilizado.

O quarto capítulo mostra as principais diferenças entre as duas tecnologias e as dificuldades encontradas durante as implementações do modelo para reservas de hotéis.

Por fim, o quinto capítulo vem para inclui as conclusões do trabalho e propostas futuras.

Capítulo 2

Tecnologias e Ferramentas Utilizadas

A Internet sem fio utiliza o protocolo *WAP* (*Wireless Application Protocol*) que por sua vez precisa da linguagem *WML* (*Wireless Markup Language*) para descrever seu conteúdo em mini-páginas. Como as páginas *WAP* usam *WML*, para poder visualizá-las é necessário um *browser* capaz de entender *WML*. Este *browser* está presente em todos os dispositivos de comunicação que suportam o protocolo *WAP* [Fig 00]. Também é possível aos desenvolvedores de sistemas testar suas aplicações diretamente nos diversos simuladores que estão disponíveis na *Web* [WMLClub], tais como: o *WinWap*, o *EasyPad Waptor*, o *M3Gate*, entre outros. Detalhes sobre simuladores serão vistos mais adiante neste capítulo.

A arquitetura do sistema *WAP* é muito simples. Nota-se, que diante dela, qualquer tecnologia utilizada para construção de aplicações em *Web* é facilmente adaptada para aplicações *WAP* [Fórum].

Seguindo os passos do *WAP*, no Japão, a operadora *DoCoMo*, concorrente da *KDD Communications* - *WAP*, lançou o padrão "*i-Mode*". O *i-Mode* virou um caso de sucesso da Internet sem fio. A operadora possui aproximadamente 33 milhões de usuários, contra os 5 milhões de assinantes *KDD Communications*. Graças ao sistema de comutação de pacotes, os usuários da *NTT DoCoMo* têm conexão permanente com a *Web*. Isso explica por que a lenta velocidade de transmissão de dados de 9,6 *Kbps* não afeta de forma tão direta o desempenho do *i-Mode*, como acontece com o *WAP*.

O serviço *i-Mode* necessita da linguagem *Compact HTML*, ou *cHTML*. Uma página Web escrita em *cHTML* pode ser vista normalmente por um *browser* como o *Netscape* ou *Explorer*. Os celulares *i-Mode* da *NTT DoCoMo* que possuem *minibrowsers cHTML*, já suportam gráficos, *GIFs* animadas e arquivos *MIDI* para *download*.

Da mesma forma que existem simuladores para *WAP*, existem também simuladores para *i-Mode*. Alguns nomes são: o *p501i*, o *Wapprofit iMode Editor* e o *i-browser* etc [Palow].

A *W3C* (*Wide World Web Consortium*), entidade que coordena o desenvolvimento e a manutenção dos padrões e demais assuntos relativos à Internet, oficializou, em fevereiro de 1998, as especificações de uma linguagem chamada *XML* (*Extensible Markup Language*). A partir daí, surgiram algumas novas linguagens baseadas na *XML*. Uma delas é a já famosa *WML*. Outra é a *XHTML* (*eXtensible HTML*) que deverá substituir gradativamente a boa e velha que se encontra na versão 4.0.

O *WAP Forum*, visando uma convergência com os padrões estabelecidos pelo *W3C*, especificou uma versão reduzida da *XHTML*, chamada *XHTML BASIC*, para uso em dispositivos móveis. A linguagem *XHTML BASIC*, representa, de certo modo, o fim da "guerra" entre o *WAP* e o *i-Mode*. Pelo menos, o fim de uma das batalhas: a das linguagens (*WML/WAP versus cHTML/i-Mode*). O termo *i-Mode* passou a designar, na mídia, tanto a tecnologia quanto o próprio aparelho celular da mega empresa japonesa *NTT DoCoMo*. O *WAP* e o *i-Mode* competem hoje como tecnologia de acesso à Internet via dispositivos móveis. A *NTT DoCoMo* e o *WAP Forum* concordaram em adotar esta nova linguagem como padrão para o futuro da *Wireless Web*.

Hoje temos a nova geração do *WAP*, chamada *WAP-NG* (*WAP New Generation*). Trata-se do *WAP 2.0*, tido como um enorme passo visando o estabelecimento de um padrão inicial para a *3G* (Terceira Geração). A linguagem para construção das mini-páginas *WAP* versão 2.0 é o *XHTML BASIC*. No Japão, a *3G* lançada pela *NTT DoCoMo*, com infra-estrutura *NEC/Ericsson/Fujitsu* utiliza a tecnologia de transmissão *WCDMA* (*Wideband CDMA*) e, do mesmo modo que o *WAP*, utiliza a linguagem *XHTML BASIC* para construção das mini-páginas que serão exibidas pelos celulares *i-Mode*. O *WAP* pode trabalhar com *CDMA*, *TDMA*, *GSM*, *GSM* com *GPRS*, *PDC*, *PHS*, *FLEX*, *ReFLEX*, *iDEN*, *TETRA*, *DataTac*, *Mobiltex* e *CDPD* (*Cellular Digital Packet Data*).

2.1 *Wireless*

O que é *Wireless*? Na concepção da palavra *Wireless* quer dizer "sem fio". Conceitualmente podemos definir *Wireless* como uma tecnologia que disponibiliza a transmissão de dados, som e imagem via ondas de rádio em freqüências superiores a 800 *MHz*. Este tipo de tecnologia pode nos propiciar diversas funcionalidades tais como: consultar qualquer tipo de notícias; confirmar reserva de restaurantes ou hotéis; confirmar horários de vôos; verificar condições de tempo e trânsito; checar *e-mails*; pagar hospedagens e outras

compras feitas num shopping, e tudo isso em qualquer lugar ou a qualquer hora, através de dispositivos móveis. Em resumo podemos dizer que a tecnologia *Wireless* vem disponibilizar a portabilidade da informação em qualquer lugar.

Falando sobre as aplicações possíveis utilizando a tecnologia *Wireless* juntamente com os *PDA's* o campo de desenvolvimento é extremamente fértil. Diversos são os tipos de aplicações que podemos imaginar, desde soluções para rede de lojas de varejo e *fast food* passando logística de materiais, a *software* para *agro-business*.

2.2 *WAP - Wireless Application Protocol*

Em junho de 1997, quatro empresas - *Nokia*, *Ericsson*, *Motorola* e *Phone.com*, uniram-se para desenvolver um protocolo de aplicações sem fio em comum, o protocolo *WAP* [Fórum]. A iniciativa teve o objetivo de unir os esforços das companhias para portar aplicações avançadas e conteúdos de Internet para dispositivos móveis, tais como telefones celulares, *pagers*, *PDAs*, entre outros.

A arquitetura Internet *World Wide Web* proporciona um modelo de programação poderoso, como pode ser visto na Figura 2.1, onde está representado o modelo de programação *WWW* [Fórum]. O conteúdo e as aplicações são apresentados em um formato padrão de dados, e visualizados através de um *Web browser*. O *Web browser* é uma aplicação de rede, ele envia requisições de objetos de dados para um servidor de rede, e este servidor responde com os dados codificados, usando um formato padrão [Les01].

O modelo de programação *WAP* é similar ao modelo *WWW*. Ele está representado na Figura 2.2 conforme [Fórum]. Também oferece muitos benefícios para os desenvolvedores de aplicação, pois o modelo de programação já é familiar. A arquitetura também é similar, e a utilização de ferramentas já existentes (por exemplo, *Web Server*, ferramentas *XML*, linguagens de *script*, etc).

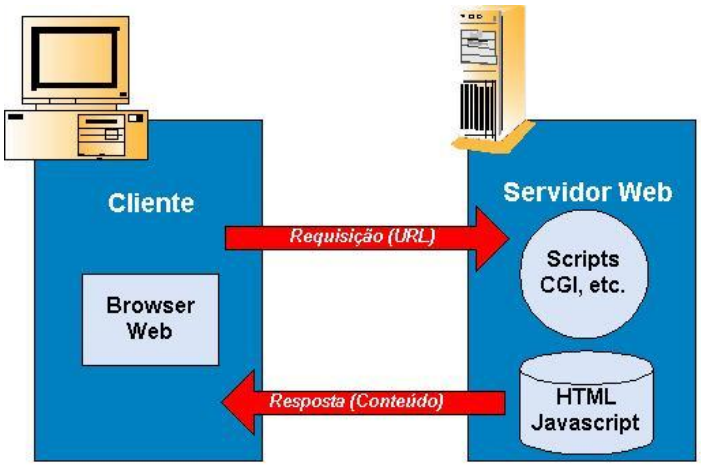


Figura 2.1 – Modelo de Programação *World Wide Web*

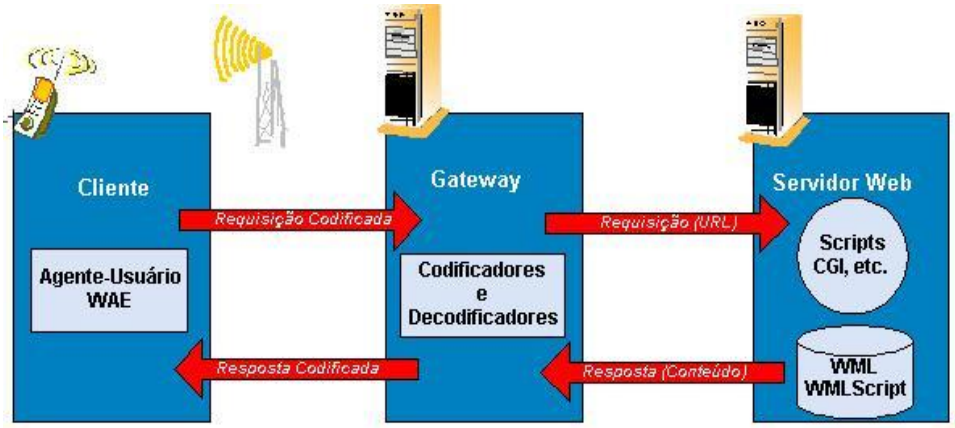


Figura 2.2 – Modelo de Programação *WAP*

O *WAP* especifica dois elementos essenciais para a comunicação sem fio: um protocolo de comunicação fim-a-fim e um ambiente de aplicação baseado em visualizadores *"browsers"* [Spo00]. Basicamente o protocolo *WAP* é uma pilha de protocolos de comunicação que tem como meta 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.

2.2.1 As camadas do WAP

A figura abaixo apresenta um comparativo entre os padrões *WAP* e *Web*. Observe a composição das camadas do protocolo *WAP*.

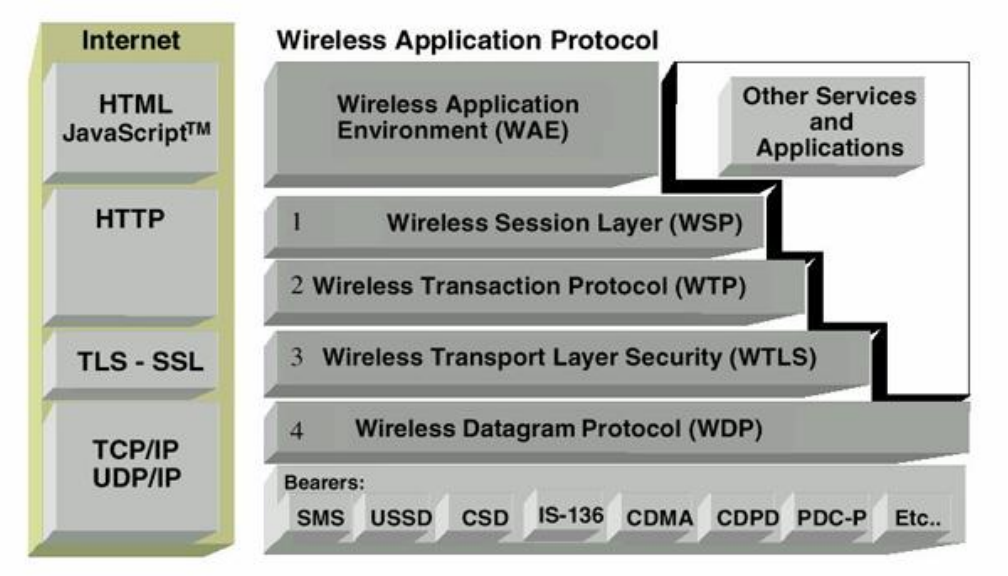


Figura 2.3 – Protocolo *WAP*

Basicamente podemos observar as seguintes entidades [Spo00]:

- *Wireless Application Enviroment* (*WAE*): Consiste na camada de "aplicação" do protocolo, estabelecendo padrões para os *browsers* (visualizadores) e para a linguagem de "renderização" (*WML*). Na Internet seria equivalente à especificação do *HTML*.
- *Wireless Application Protocol* (*WAP*): Divido em 4 subcamadas (1-Sessão, 2-Transação, 3-Segurança e 4-Datagrama) é a camada responsável pela requisição e transporte dos dados, implementando neste nível requisitos de segurança e controle transacional. Seu equivalente na Internet seria uma mistura de *HTTP* com o *TCP/UDP*.
- *Bearer*: É a camada equivalente ao nível de enlace da arquitetura *OSI/ISO*, responsável pela condução dos dados "no ar", entre a celular da operadora até o dispositivo em wireless, varias tecnologias podem ser empregadas, comumente se usa *SMS* e *CSD*.

O ambiente de aplicação (*WAE*) segue a filosofia de um *browser Web*. Dentro dos dispositivos móveis (sem fio), há um *microbrowser*, que interpreta dados recebidos de um servidor, tendo a funcionalidade de executá-los, no caso

de um *script* e/ou de renderizá-los no *display* do aparelho, ou seja, ele faz a interface com o usuário. Tal *browser* é bastante semelhante ao *Web Browser*, porém, ele apenas interpreta e renderiza código *WML*. Detalhes da linguagem *WML* e exemplos de *microbrowsers* podem ser vistos respectivamente nos itens 2.5.1 e 2.7 deste capítulo.

Segundo [Spo00], a arquitetura do sistema que possibilita que o dispositivo móvel (comumente um celular) se conecte com a Internet, em si, é muito simples. Basicamente o dispositivo móvel estabelece um circuito de troca de dados (*CSD - Circuit Switch Data*) junto ao hardware da operadora (na verdade, o celular liga para um modem e estabelece uma linha de dados e não de voz). Após isso, o dispositivo está pronto para fazer requisições à rede, entretanto ele não faz essa requisição diretamente, faz isso através de um elemento que desempenhará o papel de *proxy* de sua conexão. Esta entidade, geralmente um simples computador, é conhecida como *WAP Gateway* e tem a função de fazer as requisições *HTTP* na Internet pelo celular.

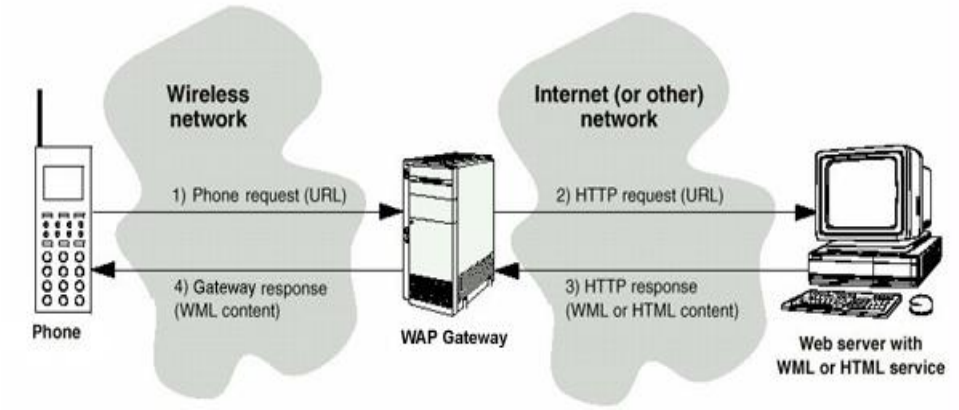


Figura 2.4 – Arquitetura do Sistema *Wireless*

2.2.2 Linguagens usadas na *Web* e no *WAP*

Na *Web*, os *sites* são constituídos de páginas com extensão "*html*", sendo sua primeira página instituída como *Home page*. Na rede *Wireless*, a extensão "*wml*" é a linguagem de marcação utilizada para exibir documentos em dispositivos compatíveis, o *mini-site* é chamado de *deck* (maço, grupo) e a mini-página é chamada de *card* (acompanhe na Tabela 2.1). Um *site* é um conjunto de páginas, enquanto um *deck* é um conjunto de *cards*. Mas as telas dos celulares estão crescendo, já podem ser roladas.

	WEB	*WAP*
Linguagem	*HTML*	*WML*
Conteúdos dispostos	*Site*	*Deck*
Página	*Home page*	*Card*

Tabela 2.1 - Linguagens usadas na *Web* e no *WAP*

Novos termos também estão surgindo, dentre eles: "*mini-sites*" ou "*WAP sites*", "*mini-pages*" ou "*WAP pages*", "*WAP phones*" (celulares *WAP*), "*WAP browsers*" etc.

2.2.3 Aplicação *WAP*

A aplicação *WAP* consiste em um servidor de aplicação e um cliente *Wireless* que faz *downloads*, através de um *gateway*, do servidor para o cliente de dados para exibição e de *scripts* para a execução [Spo00]. Semelhante ao sistema *Web* na Internet. A plataforma *WAP* prove padrões para a consistência entre *browsers* e *script interpreters*. O *microbrowser* é muito similar ao da *Web* e pode manipular conteúdos descritos na especificação do *WML*.

O *WMLScript Interpreter*, presente no *microbrowser*, permite que trechos de códigos sejam executados no dispositivo sem fio. Também estende um pouco a implementação de *scripts* permitindo a implementação de conjuntos de bibliotecas que permitem acesso a serviços do dispositivo *Wirelesss*. Tanto *WML* como o *WMLScript* foram especificamente projetados para uso com dispositivos sem fio, com pequena banda de passagem de rede e, ambos são compilados em códigos binários para otimizar a eficiência de transmissão por microondas entre as estações e os dispositivos sem fio.

As etapas para a realização da busca por uma URL de um dispositivo sem fio na arquitetura *WAP* podem ser vistas na Figura 2.5 [Spo00]:

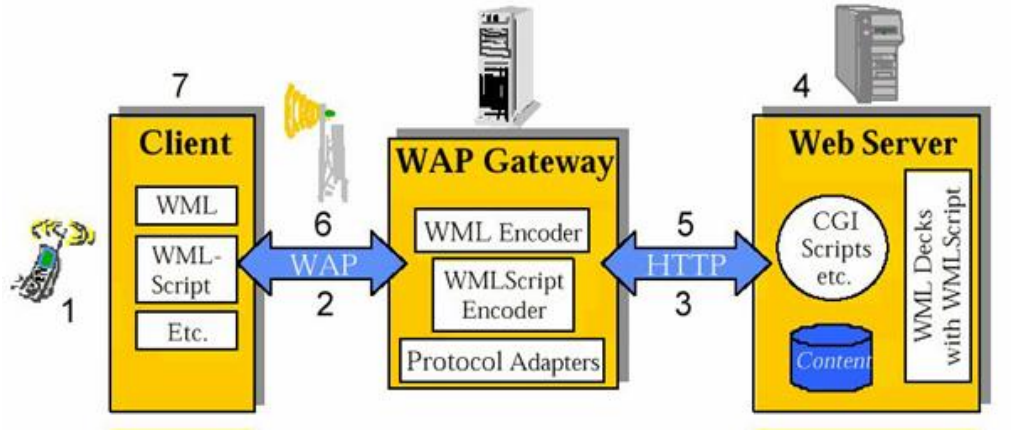


Figura 2.5 - Etapas para a realização da busca por uma URL

1. Usuário pressiona uma tecla do aparelho celular para que ele faça a requisição de uma URL específica.

2. O dispositivo então manda a requisição de URL para o *WAP Gateway*, usando para isso a pilha de protocolos *WAP*.

3. O *WAP Gateway* cria então um *HTTP* convencional acessando o *Web Server* (ou qualquer outro serviço disponível por *HTTP*) especificado pela URL.

4. No *Web Server*, a requisição *HTTP* é processada, seja ela um *CGI-BIN*, um *ASP Script*, um *Java Servlet* executando as instruções dadas pela URL.

5. O *Web Server* retorna então um *WML*, informando no *header* da resposta que o conteúdo é do tipo *'text/vnd-wap-wml*' em vez do habitual *'text/html'*. Este código pode ser estático ou gerado dinamicamente por um *script*.

6. O *WAP Gateway*, recebe a resposta, então verifica o *header* da mesma e o conteúdo *WML* recebido, compilando-os em um formato binário. Depois disso, cria uma resposta *WAP* e envia de volta ao dispositivo sem fio.

É fácil notar, que diante desta arquitetura, qualquer tecnologia utilizada para construção de aplicações em *Web* é facilmente adaptada para implementações de aplicações *WAP*, uma vez que a arquitetura Cliente/Servidor naturalmente isola a plataforma do servidor e o *WAP Gateway* acessa os serviços utilizando o mesmo protocolo *HTTP*. Do ponto de vista estrutural a diferença está no conteúdo que o *Web Server* fornecerá, que agora, em vez de *HTML* será *WML*.

Entretanto a transição *Web* para *WAP* não é trivial. Uma aplicação *WAP/WML* é "ordens de grandeza" mais limitada (em todos os aspectos) do que uma aplicação *Web/HTML*. A limitação do tamanho do conteúdo, da velocidade, da interface de entrada de dados (geralmente um teclado telefônico) e da interface de saída (uma tela LCD de 3 x 2 cm) forçam que tais aplicações sejam muito bem estudadas e trabalhadas para permitir que o usuário interaja sem grandes dificuldades. Apesar dessas limitações o poder de acessar informações de qualquer lugar e em qualquer momento torna uma aplicação *WAP* poderosíssima para domínios onde se faz presente tal necessidade.

2.3 *i-Mode*

2.3.1 O que é o *i-Mode*?

O *i-Mode* é um serviço de acesso à Internet desenvolvido pela operadora japonesa *NTT DoCoMo* e oferecido para a telefonia celular no Japão desde fevereiro de 1999 [Eurotec]. Com *i-Mode*, mais de 33 milhões de japoneses (cerca de 25% da população) têm acesso a informações, à Internet, a *e-mail* e a serviços diversos. A taxa de crescimento chega a surpreendente marca dos 28.000 novos assinantes por dia. Acompanhe este crescimento na Figura 2.6. Tamanho resultado desse sucesso é devido a três vertentes: tecnologia da rede, preço e criatividade, além, é claro, das razões culturais e históricas; o Japão foi o primeiro país a implementar um sistema telefônico celular de alta capacidade, em 1979 na cidade de Tóquio, cerca de quatro anos antes da primeira experiência americana, em Chicago.

Calcula-se que há hoje mais de 40 mil *sites i-Mode* com conteúdos diversos [Eurotec], além de serviços especializados tais como operações financeiras em bancos, reserva de passagens aéreas, pagamentos de cartão de crédito, hotéis, shows, compra de livros, orientação sobre restaurantes, entre outros. O serviço pode ser acessado de qualquer parte do Japão. As taxas de utilização do serviço são facilmente compreensíveis pelo usuário, pois é cobrado o volume de informação acessado (pacotes de informação), e não o tempo de utilização do serviço. Detalhes serão vistos mais adiante.

Com o *i-Mode*, é possível enviar e receber *e-mails*, tanto para outros usuários de *i-Mode* como para usuários "normais" da Internet [Eurotec]. A tecnologia da DoCoMo também tem um engenhoso sistema de informações locais. A empresa sabe a região onde o usuário está, rastreando a repetidora (célula) que ele está usando. Logo, é possível pedir informações de restaurantes, hospitais e até números de telefones úteis de acordo com a região. Pelo *i-Mode* o usuário pode, por exemplo, descobrir onde fica o *McDonald's* mais próximo ou acionar o *Traffic Report* e descobrir a melhor rota de escape de um congestionamento. Tudo em tempo real.

A Figura 2.6 mostra como foi o crescimento do uso do *i-Mode* desde seu surgimento em 1999 até julho deste ano [Eurotec]. O gráfico demonstra o tamanho sucesso adquirido ao longo destes anos. Observe nesta figura que em janeiro de 2001 já se podia contar com novos serviços baseados na linguagem *Java*. O uso de celulares que ofereciam estes novos serviços cresceu de maneira similar à popularidade do *i-Mode*.

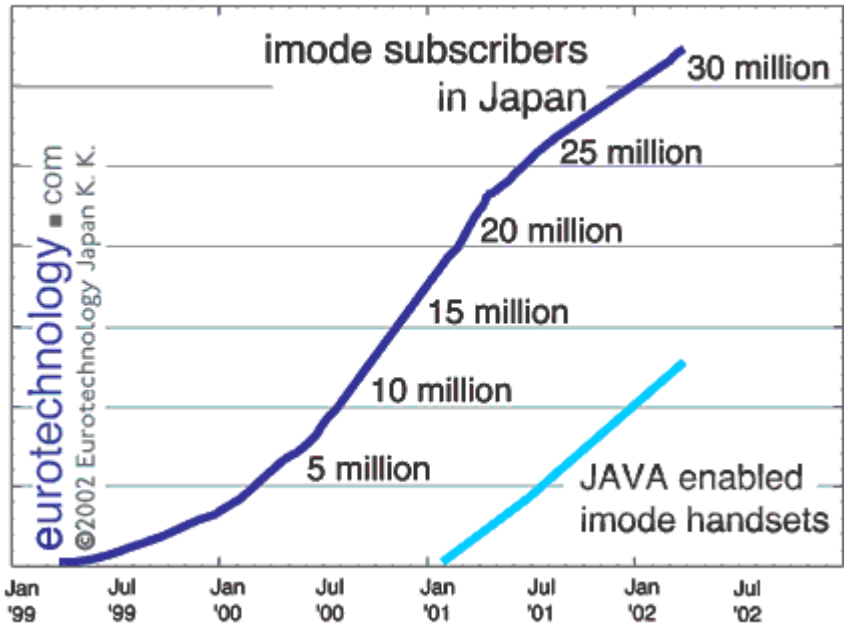


Figura 2.6 – Número de usuários utilizando o *i-Mode*.

2.3.2 Como funciona?

É basicamente um serviço de informação por pacotes. Os engenheiros de *DoCoMo* idealizaram uma rede de pacotes comutados ao longo da rede digital e celular da empresa. Com este sistema de informação "em pacotes", diferentemente das redes telefônicas de comutação de circuitos, não é necessário que cada usuário receba a informação através de um só canal de rádio, o que significa que um grande número de pessoas pode ter acesso à informação simultaneamente. Além disso, o modelo em pacotes ajuda a reduzir os custos, já que as tarifas baseiam-se no volume de informação enviada e recebida [Eurotec].

Acompanhe a seguir o funcionamento da rede *i-Mode* durante uma transmissão.

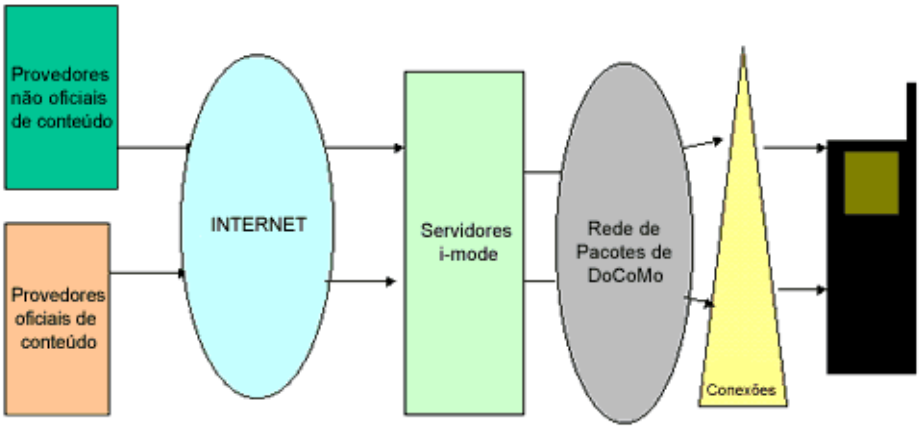


Figura 2.7 – Funcionamento da rede *i-Mode*

- Provedores: Os provedores de conteúdo estão divididos em: *Oficiais* e *Não-oficiais*. Atualmente há cerca de 300 provedores *Oficiais* de conteúdo que proporcionam *e-mail* e *Chat*, jogos, horóscopos *on-line*, calendários e boletins de notícias personalizados. Segundo [Yu01], um dos *sites* mais populares entre os jovens japoneses, o *Bandai*, tem mais de 1 milhão de assinantes, que pagam no mínimo 1 dólar de assinatura mensalmente. Ele é apenas um entre os 22.000 *sites* com conteúdo exclusivo para o *i-Mode*.

- Servidores *i-Mode*: Os servidores *i-Mode* realizam uma divisão da informação em bases de dados adaptadas ao perfil do usuário, o que permite que estes tenham um acesso mais fácil à informação. Estes

servidores realizam também o faturamento por volume de informação enviada e recebida. Igualmente, devido ao fato de que os provedores de informação podem "impulsionar" o envio de informação, os servidores *i-Mode* podem comprovar se este tipo de informação ajusta-se ao tipo de informação à que esteja registrado o usuário, evitando o envio de informação descartável.

- Rede de pacotes: Uma mensagem é dividida em pequenas partes conhecidas como pacotes ou "*packets*". Cada pacote conduz um trecho da mensagem e também informação sobre sua origem, destino e como se ligar aos pacotes vizinhos; se chegarem fora de ordem ao receptor, a mensagem pode ser facilmente e instantaneamente reconstruída. O acesso é feito através de um portal em que os provedores de conteúdo credenciados pela *NTT DoCoMo* são escolhidos a partir de um menu. Mas já existem mais de 4 mil páginas de outros provedores que são acessadas a partir da digitação do endereço *URL* no teclado do celular *i-Mode*.

- "Conexão permanente": No *i-Mode* "disca-se" apenas uma vez e a conexão se torna permanente enquanto houver sinal e carga na bateria. O acesso "persistente" é proporcionado pela tecnologia de comutação por pacotes, semelhante à da *Web* "com fio". A conexão inicial é feita em poucos segundos; sendo estabelecida, está pronta para uma imediata transferência de dados; as "reconexões" subseqüentes serão feitas em milisegundos: daí a sensação de uma conexão "permanente". O sistema utiliza compressão para aumentar o volume de dados transmitidos e o resultado final é o uso eficiente da faixa de freqüência. Apesar de "sempre ligado", o usuário paga (uma tarifa razoavelmente pequena) somente quando envia ou recebe mensagens, ou seja, a cobrança de tarifas é pelo volume de dados e não pelo tempo de conexão.

- Velocidade de transmissão: A velocidade de transmissão é de 9,6 *kbps* (kilobits por segundo). Os críticos e concorrentes acham muito pouco, mas isso não chega a ser empecilho já que tecnologia proporciona uma compressão de dados que aumenta o volume transmitido. *E-mails* e pequenos gráficos trafegam tranqüilamente nesta velocidade. E os japoneses, sempre muito práticos, estão mais preocupados com o conteúdo das mensagens do que com a sofisticação visual da tela do dispositivo.

2.3.3 Linguagem utilizada nas mini-páginas

As mini-páginas ou *mini-pages* são escritas em "*compact HTML*" (cHTML), um sub-conjunto da *HTML 2.0*, *3.2* e *4.0* (Veja a descrição desta linguagem e exemplos no item 2.5.2). Assim, uma página *Web* escrita em *cHTML* pode ser vista normalmente por um *browser* como o *Netscape* ou *Explorer*. Além disso, utiliza-se o mesmo protocolo da Internet e os provedores não precisam reconfigurar seus servidores. A adaptação de uma página *HTML* é uma tarefa simples comparada à programação *WML* do serviço *WAP* [Eurotec].

2.3.4 Quais são as razões para seu sucesso?

São várias as razões do sucesso do *i-Mode*. Em primeiro lugar, é o primeiro serviço que oferece acesso constante a Internet com a única limitação da duração da bateria. Outra grande vantagem para o usuário é o baixo preço de registro e, sobretudo a vantagem de que somente se paga pelos dados enviados e recebidos e pelos serviços incluídos à livre escolha e que normalmente são oferecidos pelos provedores *oficiais*. Diferentemente do Brasil, onde as operadoras cobram por *air-time*, ou tempo de uso, no modelo japonês cobra-se apenas pela quantidade de dados transmitida. Cada 128 bytes de informação custam para os japoneses apenas 0,03 *iene*, ou 0,0005 real! Ou seja, a cada 1.000 pacotes de informação, que suportam até 16.000 caracteres - o que pode significar 64 *e-mails* com 250 caracteres cada (que é o limite do protocolo) -, os clientes da *NTT DoCoMo* pagam 50 centavos de real [Yu01].

Estas prestações, junto com a cobertura de cerca de 98% do território japonês tornaram possível que aproximadamente 10 milhões de japoneses estejam registrados nesse serviço. Calcula-se que dos 20 milhões de usuários de Internet que há no Japão, somente cerca de 3 ou 4 milhões tenham acesso desde sua casa. Os registros diários a este serviço superam as 20.000 pessoas [Yu01].

Uma peculiaridade do mesmo consiste em que o faturamento a ser realizado é feito pelo operador que é quem possui os dados do cliente. Os provedores de conteúdos com serviços de cobrança o fazem a partir da fatura emitida pela operadora ficando esta com 9% da quantidade faturada.

Diversas pesquisas evidenciaram que os serviços mais utilizados pelos usuários deste serviço são *e-mail*, banca móvel, horários dos trens e informação de transporte em geral. Estes serviços são oferecidos pela Operadora com bastante êxito e não exigem alta velocidade de transmissão.

Outra característica igualmente vantajosa é a importância que se dá à comunicação por voz. Ou seja, se recebemos uma chamada enquanto estamos navegando passa-se imediatamente à chamada e, quando esta termina, volta-se automaticamente à tela do *i-Mode* a conexão a qual tínhamos acessado antes de

receber a chamada.

Conhecer as bases do sucesso do *i-Mode* na Ásia é importante porque será muito similar ao sucesso da terceira geração no mundo e a *DoCoMo* utiliza-se desta técnica como um grande referencial para ter idéias de futuros negócios.

2.3.5 Quais são suas limitações?

O serviço atual tem uma velocidade de transmissão de 9.6 *kbps*, algo lento hoje em dia, e mais ainda quando pensamos nas velocidades de transmissão que permitem as redes de terceira geração (*3G*). Essa baixa taxa de transmissão faz com que o protocolo japonês esbarre na dificuldade de transportar imagens, assim como o *WAP*. Além disso, a *DoCoMo* anda sofrendo forte concorrência no seu próprio país, já que seus dois maiores rivais, a *DDI* e a *IDO*, também possuem um serviço de transmissão de pacotes, porém operando numa velocidade maior, 14.4 *kbps*. Veja o item 2.4.7 "A concorrência". Outra limitação é a descarga de imagens a dois *frames*, valor muito baixo, dizem os analistas de tecnologia. Enquanto que o *e-mail* está limitado a uns 250 caracteres por mensagem, algo muito limitado se comparado ao WAP.

2.3.6 O que a *NTT DoCoMo* tem feito para melhorar seu serviço?

Em avançado estágio de desenvolvimento, principalmente no Japão, o atual conceito de células geográficas que deu origem ao popular "celular", deverá ser expandido para células de tamanhos diferentes (pico, micro e macro-células); cada célula deverá adotar uma tecnologia diferente para a "interface aérea" ou transmissão dos sinais de rádio; as macro-células provavelmente terão como base a estrutura do *GSM* "*Global System for Mobile Communications*" na tecnologia *W-CDMA* "*Wideband Code Division Multiple Access*" criada pelos militares norte-americanos; velocidades elevadas de transmissão e integração completa de tecnologias de telecomunicações e eletrônica de estado sólido servirão de suporte para sistemas multimídia com enorme poder de processamento. A *ITU* "*International Telecommunication Union*" está ultimando as especificações de um novo padrão para os sistemas móveis de terceira geração "*3G*" já conhecido como *IMT-2000* [Ros00]. A *NTT DoCoMo* espera liderar o mundo na evolução do atual sistema para a nova geração de celulares, a *3G* que permite transmissões de 386 *Kbps* e o envio de imagens de vídeo. A *NTT DoCoMo* é detentora de parte da patente da *W-CDMA*, junto com a *Qualcomm* e a *Lucent*.

Vale ainda lembrar que em outubro de 2001, a companhia *NTT DoCoMo* se uniu à americana *Packet Video Corporation* - da qual fazem parte, gigantes como a *Motorola*, a *Philips*, a *Intel* e a *Sony*, entre outras e juntas formaram a *FOMA*. A mega fusão visa o investimento pesado em desenvolvimento de

aplicações para telefonia celular de terceira geração. São 32 empresas, entre elas *IBM*, *PriceWaterHouseCoopers* e *Mitsubishi*, trabalhando na criação de serviços multimídia para os mais diversos setores, incluindo educação, medicina, turismo, aviação, consultoria, entretenimento e conteúdo de mídia.

2.3.7 A concorrência

De acordo com [Ros01], a concorrência japonesa da *NTT DoCoMo* é constituída basicamente pelas companhias *DDI Corporation*, *KDD corporation*, *IDO* "*Nippon Ido Tsuchin*" e *Japan Telecom Co Ltd*. As três primeiras firmaram um protocolo de intenções para uma fusão em outubro de 2000 com propósitos para poderem competir melhor com a gigante *NTT DoCoMO*. *DDI* e *IDO* também possuem um serviço de Internet baseado na transmissão de pacotes chamado *cdmaONE* que opera numa velocidade maior (14.4 kbps). Como o próprio nome diz, é um sistema compatível com o *CDMA*, mas não é ainda o *W-CDMA* que vai possibilitar a *3G*. O *gateway* "*EZ-server*" é compatível com *WAP*, mas a linguagem utilizada não é *WML*; conta com mais de 100 "*web sites*" no Japão. Estes serviços possuem nomes diferentes nas companhias: A *DDI Corporation* usa a marca *EZweb* e a *IDO* a marca *EZaccess*. Estas duas operadoras distribuem aos seus usuários, gratuitamente, uma ferramenta de desenvolvimento chamada "*My Deck Editor*", para popularizar e tornar mais atraente o serviço. Este editor permite a criação de mini-páginas em *HDML*. Esta linguagem, desenvolvida inicialmente pela *Unwired Planet*, hoje *Phone.com*, deu origem à *WML* do padrão *WAP*. O serviço de acesso à Internet da *J-Phone*, unidade de telefonia móvel da *Japan Telecom Co Ltd* chama-se *J-Sky*.

2.4 Linguagens de Marcação

2.4.1 *WML*

A especificação do protocolo *WAP* define a *WML - Wireless Markup Language*, linguagem de programação baseada no *XML*, que é utilizada para especificar o conteúdo e a interface de usuário para equipamentos que têm pouca banda de transmissão de dados e limitações de processamento e de memória, como os telefones celulares e *PDAs*. Um atual *WML 1.1* está disponível em: http://www.wapforum.org./DTD/WML 1.1.xml. Neste tutorial são apresentados o *WML* básico, exemplos demonstrando eventos e navegação, algumas *tags* definidas pelo *XML* e a maneira como estas *tags* devem ser agrupadas num documento *WML*.

2.4.1.1 Funcionalidades do *WML*

A *WML* inclui quatro áreas de funcionalidades [Les01]:

- Apresentação do texto e *layout*: a *WML* inclui suporte para texto e imagem, incluindo uma variedade de comandos para formatação e definição de *layout*.
- Organização em *Deck*/*Card*: todas as informações em *WML* são organizadas dentro de coleções de *cards* (cartas) e *decks* (grupo de cartas, baralho). As *cards* especificam uma ou mais unidades de interação com o usuário (por exemplo, um menu, ou uma tela de texto). Logicamente, um usuário navega entre uma série de *cards WML*, visualizando o conteúdo de cada uma, entrando com as informações requisitadas, fazendo escolhas, e movendo para outra *card*. As *cards* são agrupadas dentro de *decks*. Um *deck WML* é similar a uma página *HTML* que é identificada por uma *URL* e é uma unidade de transmissão de conteúdo.
- Navegação entre *cards* e *links*: a *WML* inclui suporte para a navegação explicita entre *cards* e *decks*. A linguagem *WML* também inclui provisão para o tratamento de eventos que ocorrem no equipamento, com propósito para utilizar na navegação ou mesmo para executar algum *script*. A *WML* também suporta *links*, âncoras similares aos que existem no *HTML*.
- Parametrização de *String* e gerenciamento de estados: Todos os *decks WML* podem ser parametrizados usando um modelo de estado. Variáveis podem ser usadas com "*strings*" e serem substituídas em tempo de execução. Essa parametrização faz com que fique mais eficiente a utilização dos recursos da rede.

A *WML* assume a mesma referência da arquitetura *HTML* e da *WWW* [Fórum]. O conteúdo é nomeado usando *URLs* e é completo com outros protocolos padrão que tem a semântica do *HTTP*, como o *WSP*.

Na *WML*, as *URLs* são utilizadas em duas situações: para fazer a navegação (*hyperlink*) ou para utilizar um recurso externo (uma imagem ou um *script*).

2.4.1.2 Sintaxe *WML*

Para se escrever um documento *WML* deve-se seguir algumas padronizações, assim como no *HTML*.

Um documento *WML* começa sempre com a referência ao documento *XML* que foi definido pelo *WapForum*. Esta referência é conhecida como um *prolog*. Quando o *prolog* for opcional, ele consistirá em duas linhas de código:

A declaração *XML* (que define a versão do *XML*) e a declaração de documentação (um ponteiro indicando onde se encontram as definições do documento *XML*). Veja um *prolog* abaixo:

<?xml version="1.0"?>
<!DOCTYPE WML PUBLIC "-//WAPFORUM//DTD WML 1.1//EN" "http://www.wapforum. org/DTD/WML_1.1.xml">

As principais definições da sintaxe para se criar uma página *WML* são as seguintes [Les01]:

- Entidade: O texto *WML* pode conter entidade de caracteres numéricos ou texto. Estas entidades especificam um caracter específico no conjunto de caracteres do documento. As entidades são usadas para especificar caracteres no conjunto de caracteres do documento que deve ser ignorado em *WML*, ou que pode ser difícil de entrar em um editor de texto. Por exemplo, o "*ampersand*" (&) é representado pela entidade nomeada *&amp*. Todas as entidades começam com um *ampersand* e terminam com um ponto-e-vírgula;
- Elementos: Elementos especificam todos as marcações (*markup*) e informações estruturais sobre um *deck WML*. Os elementos podem conter uma *tag* no início, o conteúdo e uma *tag* no final. Os elementos têm uma das duas estruturas: <*tag*> conteúdo </*tag*>; ou <*tag*/>. Os elementos vazios são representados pela *tag* <*tag*/>, significando que o elemento não tem conteúdo;
- Atributos: os atributos especificam informações adicionais que serão utilizados pelo elemento. <*tag atrib = "lessa"*/>;
- Comentário: os comentários em *WML* são definidos com a sintaxe: <*!exemplo de comentário* -->;
- Variáveis: os *cards* e *decks WML* podem ser parametrizados, utilizando variáveis como visto anteriormente. As sintaxes possíveis: *$identificador*; *$(identificador)*; *$(identificador:conversão)*. Parênteses são necessários quando o espaço em branco não indica o final da variável;
- *Case Sensitive*: a *WML* é uma linguagem *case sensitive*, ou seja, diferencia *tags* e atributos escritos de forma diferenciada. <*Hello*> é diferente de <*HELLO*>;
- Seção *CDATA*: as seções *CDATA* são usadas para sair dos blocos de texto e são válidas em qualquer *PCDATA*; por exemplo, dentro de um elemento. *<![CDATA [isso é <b> um teste]]>*;
- Erros: a especificação *XML* define o conceito de documento *XML* bem formado. Os *decks WML* que violarem a definição do documento, vão gerar um erro.

2.4.1.3 Tipos de Dados *WML*

Os tipos de dados suportados no *WML* estão listados abaixo [Les01]:

- Caracter (*CDATA*) - texto que pode conter entidades numéricas ou texto. O *CDATA* só é usado em valores de atributos; *PCDATA* - texto que pode conter numérico ou pode nomear entidades de caráter. Este texto pode conter *tags* (*PCDATA* é o "*CDATA* analisado gramaticalmente"). O *PCDATA* só é usado em elementos; *NMTOKEN* - um símbolo de nome, contendo qualquer mistura de nomes de caracteres, como definido pela especificação do *XML*.
- *Length*: O tipo *length* pode ser especificado como um inteiro que representa o número de *pixel* do "*canvas*" (tela, papel) ou como uma porcentagem do espaço horizontal ou vertical disponível.
- *Vdata*: O tipo *vdata* representa uma "*string*" que pode conter variáveis referenciadas. Este tipo só é usado em valores de atributo.
- *Flow*: o tipo *flow* representa a informação no nível do *card*. Em geral, *flow* é usado com qualquer marcação que seja incluído.
- *HREF*: O tipo *HREF* referência a um *Uniform Resource Locator* relativa ou absoluta.
- *Boolean*: O tipo *Boolean* referencia ao valor lógico que pode ser *TRUE* ou *FALSE*.
- *Number*: O tipo *Number* referencia um valor inteiro maior ou igual a zero.
- *xml:lang*: O atributo *xml:lang* especifica a linguagem natural ou formal de um elemento ou de seus atributos. O valor dos atributos é um código da linguagem de acordo com *XML*. O atributo identifica ao agente de usuário a linguagem usada no texto que pode ser apresentado ao usuário (por exemplo, o conteúdo de um elemento e valores de atributo).
- Atributos *ID* e *Class*: Todos os elementos *WML* têm dois atributos essenciais: o *id* e *class* que podem ser usados para tarefas como transformações no lado servidor. O atributo *id* provê em um elemento um único nome dentro de um *deck*. O atributo *class* se afilia em um elemento com uma ou mais classes. Múltiplos elementos podem ter uma *class* com o mesmo nome. Todos os elementos de um único *deck* com um nome de classe em comum são considerados parte da mesma *class*.
- *ContentType*: O tipo *ContentType* representa o tipo de mídia definido na *RFC2045*.

2.4.1.4 Estruturas *WML*

As seguintes estruturas fazem parte da *WML*:

- *Cards* (Cartas) e *Decks* (Baralhos): As informações da *WML* são organizadas em uma coleção de *cards* e *decks*, conforme já visto. O *card* especifica uma ou mais unidade da interação com o usuário, por exemplo, um menu de opções, uma janela de informações ou uma entrada de dados. Logicamente, um usuário navega através de uma série de *cards WML*, faz escolhas e move de um *card* para outro. Os *cards* são agrupados em *decks*. A menor unidade da *WML* que o servidor pode enviar para um *microbrowser* é um *deck.*
- *Templates*: O objeto *template* define um modelo que pode ser aplicado em todos os *cards* do mesmo *deck*; o uso desta facilidade justifica-se pela inserção de botões de navegação. Pode-se cancelar as características do *template* em um determinado *card*.
- *URLs*: O padrão de endereçamento da *WML* é o mesmo utilizado pela *HTML*, o protocolo *HTTP* traduz os endereços *WML* em requisições ao cliente, como se fossem requisições *HTML*. Existe um consenso que identificaria melhor os endereços de conteúdo *WML* e *HTML*, por exemplo, nomear os arquivos *HTML* com a extensão *html* e os arquivos *WML* com a extensão *WML*.
- *Links* Âncoras: As âncoras são ligações dinâmicas para arquivos ou imagens no servidor ou máquina local. O formato é semelhante ao modelo *WWW*, por exemplo:
  **
  <go href= "http://waptotal.com/"/>
  <go href= "/teste/teste.WML"/>

2.4.1.5 Exemplo de aplicação

A seguir temos um exemplo de aplicação bastante simples.

```
<?xml version="1.0"?>
<!DOCTYPE WML PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/WML_1.1.xml">

<WML>

  <card id="Login" title="Login">
    <do type="accept" label="Senha"> <go href="#Senha"/> </do>
    <p> Usuário:
    <select name="nome" title="Nome:">
      <option value="joão">João</option>
      <option value="Sergio">Sergio</option>
      <option value="Maria">Maria</option>
      <option value="Claudio">Claudio</option>
```

```
    </select> </p>
  </card>

  <card id="Senha" title="Senha:">
    <do type="accept" label="Resultado"> <go href="#Resultado"/> </do>
    <p> Senha: <input type="text" name="senha"/> </p>
  </card>

  <card id="Resultado" title="Resultado:">
    <p> Entrada:<br/> Nome: $(nome)<br/> Senha: $(senha)<br/>  </p>
  </card>

</WML>
```

Veja o resultado na figura abaixo:

Figura 2.8 – Página escrita em *WML* vista por 3 modelos diferentes de celulares para *WAP*.

Como você pode ver, o *prolog* deste documento contém a versão do *XML*. Depois deste, vem o elemento do documento *WML*, o *Deck*, que contém três cartões: *Login*, Senha e Resultado. Cada um destes cartões é definido usando um elemento *<card>*. Os cartões do *Login* e da Senha também definem

Eventos, eles utilizam o elemento <*do type ="accept"*> para definir o evento a ser chamado.

Quando o elemento "*accept*" é encontrado, ele é mostrado no display do celular (ou outro dispositivo) como uma opção.

Quem estiver familiarizado com a *tag anchor* <*a*> no *HTML* (ou no *cHTML*), e conhece as especificações do atributo *href*, então sabe como fazer um link no *browser* para uma âncora selecionada. O elemento *WML* <*go*> e o atributo *href* trabalham da mesma maneira. Com *HTML*, para fazer um link para outro cartão no mesmo documento, você simplesmente insere o símbolo # antes do *link*.

Por último, veja como é o uso da sintaxe *$(nome_da_variável)*, usada para substituição de variáveis em um cartão ou *deck*:

Nome: $(nome)<*br/*>
Senha: $(senha)<*br/*>

2.4.1.6 *WMLScript*

Em muitos casos, é necessário realizar um processamento na aplicação. Para fazer isso, utiliza-se a linguagem de *script WMLScript* que trabalha em conjunto com a linguagem *WML*.

A linguagem *WMLScript* é baseada na linguagem *ECMAScript* que é uma linguagem de *script* padrão para muitos *scripts*, tanto no lado do servidor como no lado do cliente. Devido a pouca banda de dados atualmente suportada pelo *WAP*, foi incluída uma especificação binária para otimização de dados pelo cliente. Também a linguagem *JavaScript* foi baseada no *ECMAScript*, fazendo com que *WMLScript* e o *JavaScript* sejam bem semelhantes, o que torna o aprendizado da *WMLScript* por um programador que conheça *JavaScript* bem fácil [Fórum].

A linguagem *WMLScript* foi implementada para ser utilizada somente no lado cliente, diferentemente do *JavaScript* e do *VBScript* que podem ser usadas no lado cliente e também no lado servidor. A utilização do processamento no lado cliente é por causa do problema de pouca banda de dados entre o cliente e o servidor. Se um processamento for feito no servidor e demorar muito para retornar o resultado, a aplicação irá abortar. Se o processamento é feito no lado cliente, então o problema de demora no resultado não existe. Assim como *JavaScript* e outras linguagens de *script*, o *WMLScript* roda em um *browser*. Diferentemente da *JavaScript*, *WMLScript* não fica dentro da linguagem de *markup* (*WML*). O código *WMLScript* fica em um arquivo próprio que é chamado de dentro de um *card WML*. A extensão do arquivo *WMLScript* é *.WMLs* e a extensão do arquivo *WML* é *.WML*. A chamada do programa

WMLScript é feita através de um *hyperlink* [Les01]. O exemplo na Figura 2.9 mostra um card *WML* que chama uma função *WMLScript*.

O importante é observar como funciona a chamada da função *calcula()* que está no arquivo *calcula.WMLs*. Na Figura 2.10, é listado o código do programa *calcula()* escrito em WMLScript.

```
<?xml version= "1.0"?> <!DOCTYPE WML PUBLIC "-//WAPF"RUM//DTD
WML 1.1//EN http://www.wapForum.org/DTD/WML_1.1.xml>
<--scriptchamador.WML-->

<WML>
        <card id= "main" title = "WMLScript"
        <p>
                O resultado do calculo "2+3" é: $(number)<br/>
                <a href= "chamada.WMLs#calcula(2+3)">
                        Calcular
                </a>
        </p>
        </card>
</WML>
```

Figura 2.9 - Exemplo de *Card WML* que Chama Função *WMLScript*

```
Extern calcula (a,b){
        var x; x = a + b;
        WMLBrowser.setVar("number",n);
        WMLBrowser.refresh(); }
```

Figura 2.10 - Código do Arquivo calcula.*WML*s

Analisando a linguagem *WMLScript*, foi identificado que programadores já familiarizados com a linguagem *JavaScript* praticamente não precisarão de nenhum treinamento especial para desenvolver programas em *WMLScript*, pois as estruturas das duas linguagens são semelhantes. É claro que o programador vai ter que ficar atento, principalmente em relação às limitações dos equipamentos móveis que rodam os programas *WMLScript*. Isso faz com que o programador tenha que modificar um pouco o seu modo de resolver alguns problemas, pois ele sempre tem que observar as limitações dos dispositivos clientes, e também tem que escrever uma aplicação que seja compatível com a maioria dos dispositivos móveis atuais que são de diferentes fabricantes, com telas de tamanho diferentes, e poder de processamento muito reduzido.

2.4.2 *cHTML*

O *cHTML* ou *HTML* Compacto pertence a um subconjunto das recomendações *HTML* 2.0, *HTML* 3.2 e *HTML* 4.0 [Kam98]. Foi especialmente projetado para suprir as exigências e limitações dos dispositivos móveis de comunicação, tais como: memória pequena, armazenamento reduzido ou inexistente, *CPU* com baixo poder de processamento, tela pequena com restrições de cores, caracteres simples e método de entrada restrito (sem teclado e mouse). Podemos dizer que o *cHTML* herda a flexibilidade e a portabilidade do *HTML* padrão.

HTML Compacto não depende do protocolo da rede. No caso típico o protocolo de transporte para *HTML* Compacto é assumido por *HTTP* sobre *TCP/IP*. De qualquer maneira, a atual rede de comunicação sem fio para telefones celulares é de baixa velocidade.

Uma página *Web* escrita em *cHTML* pode ser vista normalmente por um *browser* como o *Netscape* ou *Explorer*. Entretanto, para se construir páginas em *cHTML*, o ideal é utilizar *browsers* específicos para esta linguagem, a fim de obter um resultado mais próximo daquele visto por um dispositivo móvel (veja no item 2.7.2, *browsers* (simuladores) para *i-Mode*). A recomendação *HTML 4.0* inclui novas características adicionais. Por exemplo, *CSS* (*Cascading Style Sheets*) oferece uma grande variedade estilos de documentos.

2.4.2.1 Características do *cHTML*

O *cHTML* foi desenvolvido pela empresa japonesa *ACCESS COMPANY LTD.* No início de 1998 foi submetida ao organismo que regula as atividades da Internet, o *W3C* (*World Wide WEb Consortium*), que a aceitou. O *cHTML* utiliza o mesmo protocolo *HTTP* sobre *TCP/IP* [Kam98].

Com o *cHTML* todas as operações básicas podem ser feitas por uma combinação de quatro teclas: Cursor para a frente, cursor para trás, selecionar, e retornar à página anterior. Funções como gráficos complexos, figuras grandes e tabelas não estão disponíveis.

A seguir as características que não estão disponíveis para o *cHTML* [Kam98].

- Imagens *JPEG*
- Tabelas
- Imagens de mapa
- Múltiplas fontes e estilos de caracteres
- Fundo com imagens e cor
- *Frames*
- *Style sheet*

Definimos que *cHTML* inclui suporte a imagens de *GIF* e as mini-páginas devem ter no máximo 5 kbytes, mas o recomendado é que tenham aproximadamente 3 kbytes para um carregamento mais rápido.

Recomenda-se um limite de *buffer* para algumas funções [Kam98].

- *INPUT* (O tamanho máximo de *buffer* é de 512 bytes).
- *SELECT* (O tamanho máximo de *buffer* é 4096 bytes).

A lista completa dos *tags* suportados pelo *cHTML* e uma comparação entre as linguagens *HTML 2,0*, *HTML 3,2* e *HTML 4,0* podem ser encontradas em [Kam98].

2.4.2.2 Exemplo de aplicação

Vamos tomar como exemplo o programa abaixo. Note que a primeira linha define o tipo de documento.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0 Draft//EN">
<HTML>
  <HEAD>
    <TITLE>Main MENU</TITLE>
  </HEAD>

  <BODY>
    1<A HREF="PHONE.HTM" accesskey="1">Phone List<BR></A>
    2<A HREF="MLIST.HTM" accesskey="2">Mail List<BR></A>
    3<A HREF="MSEND.HTM" accesskey="3">Send Mail<BR></A>
    4<A HREF="NEWS.HTM" accesskey="4">News<BR></A>
    5<A HREF="SPORT.HTM" accesskey="5">Sports<BR></A>
    6<A HREF="WEATH.HTM" accesskey="6">Weather<BR></A>
    7<A HREF="AREA.HTM" accesskey="7">Area Info.<BR></A>
    8<A HREF="PICT.HTM" accesskey="8">Picture<BR></A>
  </BODY>

</HTML>
```

No entanto, esta página foi escrita para ser vista por um celular, no caso, um *i-Mode* da *NTT DoCoMo*. Veja o resultado na figura abaixo:

Figura 2.11 – Página escrita em *cHTML* vista por um celular *i-Mode*

Neste exemplo, foi utilizado o atributo chamado "*accesskey*" que permite efetuar um desvio através de um *hyperlink* utilizando-se uma tecla numérica do celular. Por exemplo, para escolher a opção *5-Sports*, não é necessário levar o cursor até o item 5, basta pressionar a tecla 5.

2.5 *Scripts Web*

Segundo [Kin01], o *script Web* é a ferramenta responsável em fazer a comunicação entre o banco de dados e o servidor *We*b. Pode ser um programa compilado como o *CGI* ou interpretado como o *PHP* e o *ASP*.

Há vários outros *scripts We*b, mas os três nomes citados acima são os mais populares. Outras alternativas poderiam ser: o *JSP (JavaServer Page*s), *Javascript*, *SSI (Server-Side Include*s), *API (Application Programming Interface*), *ColdFusion* e *Java servlets*.

Como escolha para o desenvolvimento das aplicações *WAP* e *i-Mode* decidimos fazer o uso do *PHP*.

2.5.1 *PHP*

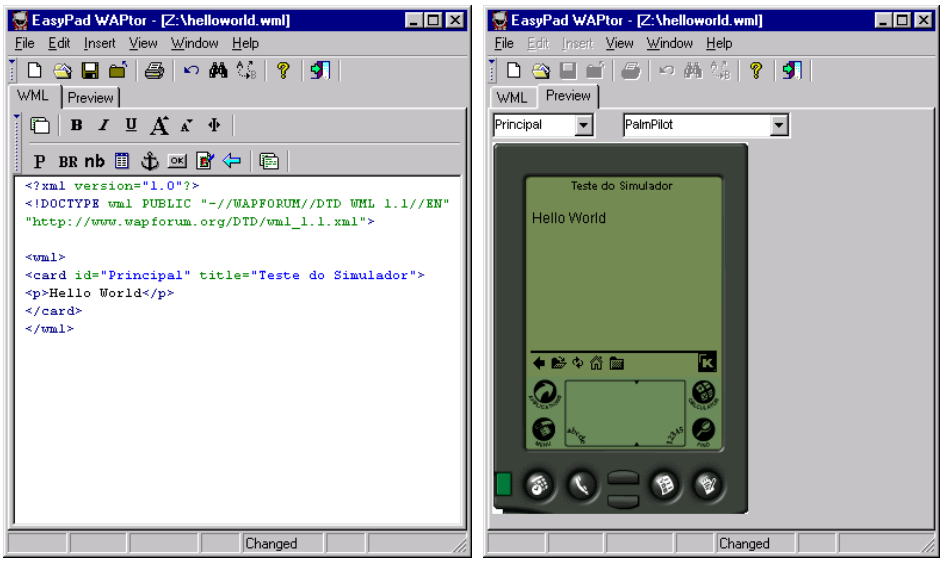
Assim como o *ASP*, o *PHP (Personal Home Page*) é um *script Web* cujo código *PHP* é executado no servidor, sendo enviado para o cliente apenas o código *HTML*, *cHTML* ou ainda *WML*, dependendo do como está a sintaxe em *PHP*. Desta maneira é possível interagir com bancos de dados e aplicações existentes no servidor, com a vantagem de não expor o código fonte para o cliente. A sintaxe do *PHP* é baseada nas linguagens *C*, *Java* e *Perl*, facilitando o desenvolvimento para pessoas familiarizadas com tais linguagens. Uma dos pontos fortes do *PHP* é o suporte a um grande número de bancos de dados, como *Informix*, *Microsoft SQL Server*, *MySQL*, *Oracle*, *Sybase*, *PostgreSQL* e vários outros. Além disso, pode ser utilizado tanto em *Windows* como em *Unix* (portabilidade) e possui código aberto.

2.6 Ferramentas para auxílio na confecção e visualização de páginas i-*Mode* e *WAP*

Editores de código e simuladores (ou emuladores) correspondem a dois tipos de ferramentas bastante úteis e necessárias no processo de confecção de mini-páginas para dispositivos móveis. Essas ferramentas geralmente possuem uma interface "amigável" e muitas delas podem ser baixadas na Internet. A seguir algumas destas ferramentas.

2.6.1 Editores de código

Antes de citar alguns dos editores de código específicos para *WAP* e *i-Mode*, é bom lembrar que existem alguns editores comuns de texto que também podem ser usados para este fim. Um exemplo típico é o Bloco de Notas. Entretanto, alguns inconvenientes são observados quando se usar estes editores, pois algumas funções, como por exemplo, correção automática de *tags*, palavras reservadas em destaque; não estarão disponíveis. Além disso, alguns editores além de funcionarem como editores de código, permitem também uma pré-visualização dos resultados, como é caso do *EasyPad WAPtor* (Figuras 2.12 e 2.13), desenvolvido para plataformas *Windows*.



Figuras 2.12 e 2.13 – Editor *EasyPad WAPtor 2.2*

O *Waptor* é o mais procurado dos editores *WML* existentes. Possui uma interface simples e amigável, com funções bem básicas, e lembra em muito os primeiros editores *HTML* de anos atrás. Suporta tabelas e imagens. Permite pré-visualização do código em *skins* dos formatos *Nokia 7110*, *Siemens S35*, *Ericsson R320*, *Ericsson R380*, *Motorola T7389*, *PalmPilot* (Figura 2.12) e Normal. É um programa gratuito e pode ser adquirido no *site* http://www.waptop.net/products.html.

Outros editores para *WML* são:

- *WmlEdit* – editor simples;
- *Zylog* – do tipo *WYSIWYG*;
- *CardONE* – dispensa qualquer conhecimento de *WML*;
- *Mpresence* – editor *on-line*, permite animações leves;
- *Adobe GoLive 5.0* – editor de código para *WAP* e *i-Mode*;
- *WAPobjects framework* – desenvolvido para especialmente para *MACs*;
- *Nokia WML Studio* – permite editar conteúdo *WML* junto com objetos *drag and drop*;
- *Santana Builder* – ambiente de desenho *off-line* permite criar aplicações *WAP*, ler informação diretamente da base de dados, acrescentar imagens ao *site*, utilizar formulários, eventos e acrescentar *scripts*.

Assim como o *Waptor*, o *Wapprofit iMode Editor* (Figura 2.14), editor *cHTML*, também permite pré-visualização dos resultados, além de trazer consigo a função de *debugger*. O *Wapprofit*, desenvolvido para plataformas *Windows*, suporta *tags*, menus, abertura e edição de arquivos. Seu arquivo de instalação está disponível em http://www.wapprofit.com/products/imodedownload.html.

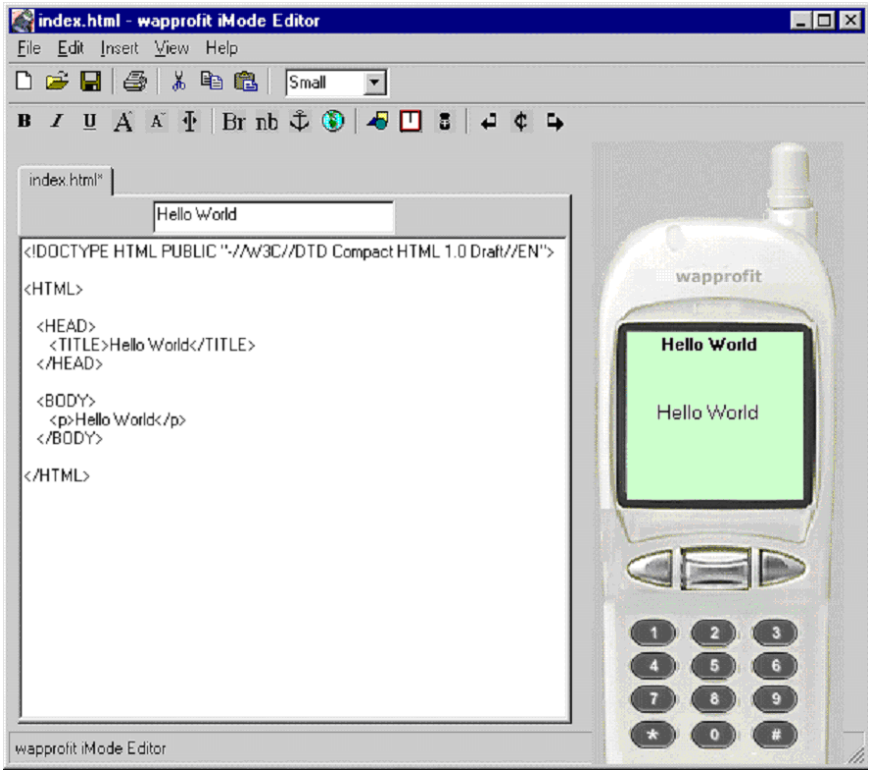


Figura 2.14 – Editor *Wapprofit iMode Editor*

Outros editores *cHTML* para *i-Mode* são:

- *Adobe GoLive 5.0;*
- *OpenWave;*
- *Modezilla;*
- *Opera.*

2.6.2 Simuladores (ou Emuladores)

Útil no processo de criação das mini-páginas, um simulador tem como função imitar o funcionamento de um celular real. Porém, algumas funções não podem ser executadas, como por exemplo, alguns ícones do *display* estão desativados e os controles de som são usados para o ajuste do tamanho das fontes. O usuário vai interagir com simulador através do teclado do microcomputador e com o cursor do *mouse* sobre as teclas do simulador.

Um dos simuladores mais conhecidos para *WAP* é o *Winwap for Windows*, disponível em http://www.winwap.org. Este simulador funciona como um *browser* que interpreta páginas em *WML* e exibe o resultado na tela do mesmo formato que o *Internet Explorer* exibe páginas em *cHTML*, ou seja, com uma aparência nem um pouco parecida com a de um celular. Apesar disso, ambos possuem a vantagem de buscar páginas via *Web*, o que não compete a outros simuladores como o *Waptor*, por exemplo.

Outros simuladores para *WML/WAP* são [WMLClub]:

- *Klondike;*
- *WAPman;*
- *ccWAP Browser;*
- *4thpass KBrowser* (*Palm*)*;*
- *WAPsilon* – converte páginas *WAP* em *HTML*
- *Opera* – agora também suportando páginas *WML*.
- *Pyweb Deck-It* – possui suporte para *Nokia 7110, Ericsson R320 e 380*.
- *Wapalizer* – simulador *on-line* disponível em http://gelon.net.
- *M3Gate* – simulador *on-line*. Possui aparência de um celular real.

Quanto à tecnologia *i-Mode*, uma vantagem pode ser observada com relação ao *WAP*. As mini-páginas por serem escritas em *cHTML* podem ser normalmente visualizadas por um *browser* que interprete a linguagem *HTML*, como por exemplo, o *Netscape* e o *Internet Explorer*. Ainda assim, diversos simuladores próprios para *i-Mode* foram desenvolvidos e alguns dos mais famosos são citados abaixo [Palow]:

- *OpenWave;*
- *i-browser* – Gratuito e fácil de usar, não necessita de instalação, pois já está no formato .exe. Funciona como um *browser* para páginas *cHTML* e algumas em *HTML*;
- *Wapprofit i-Mode Emulator* – requer para seu funcionamento do *Internet Explorer 5.0* ou de uma versão posterior. Sua versão 2.0 está disponível no *site* http://www.wapprofit.com;

- *i-Mode Emulator* – apesar de estar em japonês, ele é muito usado;
- *iTool iMode Emulator* – Emuladores *D501i/F501i*, *P501i*, *N501i* (em japonês).

2.6.3 Outras ferramentas

Existem também ferramentas desenvolvidas exclusivamente para manipulação de imagens, tais como: o *Wapdraw* usado para desenhar imagens no formato *WBMP*; o *wbmpcreator* e *Java bean* para criar imagens *WBMP* de todas as fontes; o *Pic2wbmp* especial para converter imagens feitas com *Adobe Photoshop* ou *Macromedia Fireworks* para o formato *WBMP* e *o pbmwbmp* (*Linux*), conversor de imagens *PBM* para *WBMP* [WMLClub].

Outras ferramentas desempenham a função de Cliente/Servidor, como o *WAPgate*, o *WAPLite*, o *WAPtelnet* (*Windows*, *Solaris*, *HP-UX*, *AIX*, *Linux*, *MacOS*) que funciona como cliente *Telnet* para Telemoveis *WAP*, entre outros [WMLClub].

Há também inúmeras ferramentas com funções menos importantes, porém úteis, como é o caso do *vVault Anywhere* (Telemoveis *WAP*, *Palm*), para acesso de arquivos, *faxes* e *e-mail* através de dispositivos *WAP*; e o *CitiKey* (*Windows CE*, *Palm*), para sistemas de informações geográficas urbano em *WAP*. [WMLClub]

Capítulo 3

Descrição das Implementações

As implementações baseiam-se no modelo de um *site* de reservas de hotéis elaborado por [Kin01]. Seu trabalho mostrou as vantagens e as desvantagens de três soluções que permitem o acesso de banco de dados via *Web*. Uma destas soluções integra *SGBD PostgreSQL* com *script PHP4*. Esta implementação, segundo [Kin01], mostrou-se ser a mais robusta, apresentando bons resultados na maioria das funcionalidades analisadas e por isso serviu de base para nossos estudos. O que se fez neste trabalho foi apenas modificar o modelo de [Kin01], de modo torná-lo possível de ser implementado em sistemas para dispositivos móveis, mais precisamente, sistemas para telefones celulares.

Apesar desta ligeira modificação sofrida pelo modelo original, era necessário manter o modelo modificado, um tanto quanto parecido com o original, modificando-o apenas o suficiente. Tudo isso porque um dos objetivos deste trabalho era justamente comparar o quanto seria difícil fazer esta conversão de plataformas sem efetuar grandes mudanças no modelo original desenvolvido especialmente para aplicações *Web*.

No novo modelo implementado os padrões mantidos foram: o Modelo Relacional, o Modelo Entidade-Relacionamento (ER) e as tabelas dos bancos de dados. As modificações que eram necessárias estavam a cargo da nova interface, que deveria ser elaborada para mini-telas; na digitação de um novo código que apesar de ser o mesmo do anterior, desta vez exigiu uma subdivisão de novos arquivos, seguindo as exigências da nova interface. A linguagem de *script* utilizada continuou sendo o *PHP*, da mesma forma que o *SQL* continuou sendo a linguagem para o acesso ao banco de dados. As novas aplicações foram executadas no servidor *Apache*, cuja função era de desempenhar o papel de Servidor *Web*.

3.1 Mapa do *site*

A Figura 3.1 representa o mapa do *site* “Reservas On Line” desenvolvido para telefones celulares:

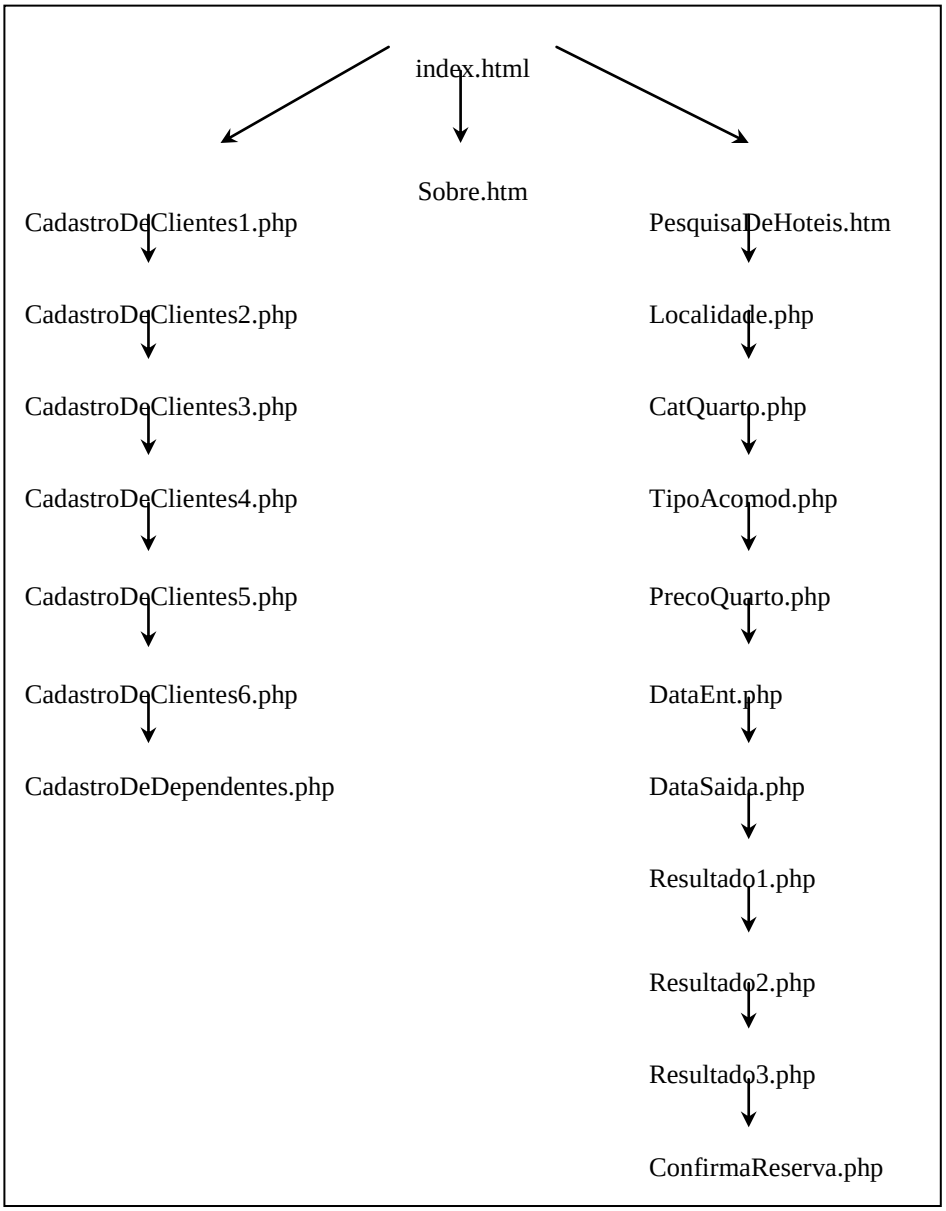


Figura 3.1 – Mapa do *site* “Reservas em On Line”

O *site* funciona da seguinte maneira:

- O Cliente se cadastra no *site* através de um formulário (clientes.html) desenvolvido em *cHTML* ou *WML*. Este formulário está subdividido em CadastroDeClientes1.php, CadastroDeClientes2.php, CadastroDeClientes3.php, CadastroDeClientes4.php, CadastroDeClientes5.php e CadastroDeClientes6.php, devido às limitações das mini-telas dos dispositivos móveis. Pelo formulário é possível cadastrar os dados do Cliente: Nome, Endereço, Telefones, Data de Nascimento, CPF e Senha. Há a opção para cadastro de dependentes. Com estes dados, o *script Web* requisita uma conexão com o banco de dados e efetiva o cadastro.
- O Cliente, depois de se cadastrar, pode fazer pesquisas customizadas, a fim de escolher em qual hotel a reserva será feita. A customização é realizada através da escolha do estado, da cidade, da categoria do quarto (*standard*, luxo ou suíte), do tipo de acomodação (solteiro, duplo, triplo ou quádruplo), do preço do quarto e das datas de entrada e de saída. Após as escolhas, o *script Web* requisita uma conexão com o banco de dados e realiza a consulta.
- O Cliente escolhe uma ou mais vagas dos hotéis que aparecem no resultado e digita a Senha e o CPF para confirmar a reserva. O *script Web*, então, efetiva o cadastro da reserva.

3.2 Etapas da Implementação

A implementação foi realizada em 5 etapas:

- A primeira etapa foi para preparar as ferramentas que seriam utilizadas nas implementações. Nesta etapa, o servidor *Apache* do *Windows 98* foi instalado e configurado para servir de plataforma de testes. Logo em seguida, criou-se o banco de dados "ReservasOnLine" e suas tabelas.
- A segunda etapa foi a de transformar a interface *Web* produzida por [Kin01] numa nova interface para ser utilizada nos telefones celulares. A nova interface pode ser encontrada no Anexo A.
- A terceira etapa utilizou-se o modelo relacional existente para a implementação em *MySQL* integrado com o *script PHP*.
- A quarta etapa foi para a implementar o código em formato *cHTML* sob o antigo código em *HTML*. O código em *cHTML* pode ser encontrado no Anexo B, e o código original (em formato *HTML*) pode ser encontrado em [Kin01].

- Na quinta, e última etapa, o código em *cHTML* foi integrado com script *PHP* para que pudesse ser processado no servidor *Apache*. Nesta mesma etapa testes foram realizados para comprovar o funcionamento do Sistema.

Por falta de tempo para concluir este trabalho, as etapas de implementação e processamento da aplicação em *WML* não foram possíveis de realizar, ficando como propostas para trabalhos futuros.

A seguir, os detalhes das etapas de implementações.

3.3 Instalação e configuração do Servidor *Apache*

O arquivo de instalação *phptriad2-2-1.exe* disponível no endereço http://phpbrasil.com instala um pacote contendo três programas: o Servidor *Apache*, o *PHP* e o *MySQL*. A instalação traz uma configuração *default* exclusiva para aplicações *HTML*.

Para *cHTML* ela também é válida, mas para aplicações *WML* algumas configurações devem ser realizadas. Dentro do arquivo *httpd.conf* presente no diretório de instalação do *Apache* o seguinte código deve ser adicionado:

```
# MIME Types for WAP
# For PHP 4.x, use this:
AddType application/x-httpd-php .wml
# For PHP 3.x, use this:
AddType application/x-httpd-php3 .wml
# For normal WML pages.
AddType text/vnd.wap.wml .wml
# For WML embedded graphics.
AddType image/vnd.wap.wbmp .wbmp
# End MIME Types for WAP
```

Após a adição deste código, o arquivo *httpd.conf* deve ser salvo para que o servidor *Apache* possa processar aplicações WML.

3.4 Criação do banco de dados

Para a criação do banco de dados e das tabelas, as seguintes instruções foram definidas:

```
1. CREATE DATABASE ReservasOnLine

2. CREATE TABLE cliente(
   cl_nome        VARCHAR(60)   NOT NULL,
```

```
   cl_rua         VARCHAR(50),
   cl_bairro      VARCHAR(25),
   cl_cidade      VARCHAR(20),
   cl_estado      CHAR(2),
   cl_cep         CHAR(9),
   cl_pais        VARCHAR(20),
   cl_cpf         CHAR(15)     NOT NULL,
   cl_dtnasc      DATE,
   cl_senha       VARCHAR(20)  NOT NULL,
   PRIMARY KEY (cl_cpf))
```

3.
```
   CREATE TABLE cliente_telefone(
   cl_tel_cpf     CHAR(15)     NOT NULL,
   cl_tel_numero  CHAR(15)     NOT NULL,
   cl_tel_tipo    VARCHAR(15),
   PRIMARY KEY (cl_tel_cpf, cl_tel_numero))
```

4.
```
   CREATE TABLE dependente(
   dep_cl_cpf     CHAR(15)     NOT NULL,
   dep_nome       VARCHAR(60)  NOT NULL,
   dep_dtnasc     DATE,
   dep_relacao    VARCHAR(40),
   PRIMARY KEY (dep_cl_cpf, dep_nome))
```

5.
```
   CREATE TABLE hotel(
   hot_nome       VARCHAR(50)  NUT NULL,
   hot_rua        VARCHAR(50),
   hot_bairro     VARCHAR(25),
   hot_cidade     VARCHAR(20),
   hot_estado     CHAR(2),
   hot_cep        VARCHAR(9),
   hot_cnpj       CHAR(15)     NUT NULL,
   hot_homepage   VARCHAR(50),
   hot_email      VARCHAR(50),
   PRIMARY KEY (hot_cnpj))
```

6.
```
   CREATE TABLE hotel_telefone(
   hot_tel_cnpj   CHAR(15)     NOT NULL,
   hot_tel_numero CHAR(15)     NOT NULL,
   hot_tel_tipo   VARCHAR(15),
   PRIMARY KEY (hot_tel_cnpj, hot_tel_numero))
```

7.
```
   CREATE TABLE quarto(
   qto_hot_cnpj   CHAR(15)     NOT NULL,
   qto_numero     INTEGER      NOT NULL,
   qto_tipo       VARCHAR(25),
   qto_preço      REAL,
   qto_nopessoas  INTEGER,
   qto_descricao  VARCHAR(20),
```

```
    PRIMARY KEY (qto_hot_cnpj, qto_numero))

8.  CREATE TABLE reservas(
    res_cl_cpf     CHAR(15)      NOT NULL,
    res_hot_cnpj   CHAR(15)      NOT NULL,
    res_noquarto   INTEGER       NOT NULL,
    res_entrada    DATE          NOT NULL,
    res_saida      DATE          NOT NULL,
    PRIMARY KEY (res_cl_cpf, res_hot_cnpj, res_noquarto))
```

3.5 Elaboração da nova interface

As categorias dos dispositivos móveis são com freqüência aparelhos comunicadores pequenos, celulares e *PDAs*, havendo portanto, algumas restrições de *hardware* que a nova interface deva atender tais como: tamanho reduzido da memória, tela pequena monocromática (preto e branco), poucas opções de fontes de caracteres (geralmente apenas uma fonte), pouca estabilidade durante a conexão e método de entrada de dados restrito, como teclado numérico.

A linguagem cHTML se assemelha muito com o HTML, com a única diferença de não permitir tabelas, múltiplos estilos de fontes, grande uso de memória, enfim, qualquer recurso que vá além das capacidades do hardware do dispositivo móvel. Por isso, a primeira medida tomada, foi retirar todos os recursos de interface que acompanhavam o modelo original do sistema de reservas de hotéis. Um novo modelo foi desenvolvido (Anexo A) trazendo consigo uma nova forma de recolher dados do Cliente para efetuar o seu cadastro. O antigo cadastro de clientes composto por um único formulário, este ocupando uma tela inteira, foi dividido em 6 novos formulários de tamanhos suficientes para serem suportados pelas mini-telas dos telefones celulares. O formulário para o preenchimento dos dependentes que antes comportava os dados de até 6 dependentes, foi transformado em um pequeno formulário com os mesmos campos, porém permitindo o cadastro de um dependente de cada vez. O formulário passou então a ser exibido para o cliente 6 vezes.

O formulário para pesquisa de hotéis do modelo original também sofreu modificações, sendo que, cada campo ganhou um formulário próprio. O campo Localidade que antes era formado por menus com opções de escolha de cidade e estado passou a ser formado de campos para digitação, assim o usuário agora deverá digitar no seu aparelho os campos correspondentes a estas opções. O mesmo aconteceu para os campos Data de Entrada e Data de Saída. Os demais campos (Categoria do Quarto, Tipo de Acomodação e Preço do Quarto) continuaram como antes.

A nova forma de exibir os resultados da pesquisa também teve que ser modificada. Não era viável mostrar todos os dados de um hotel e, logo abaixo dele, todos os quartos disponíveis. Optou-se então, por reduzir as informações referentes ao hotel acrescentado abaixo delas as informações de um determinado quarto que estivesse disponível. Caso um mesmo hotel apresente duas ou mais vagas disponíveis, o sistema exibe uma vaga de cada vez, repetindo os dados do hotel para cada quarto mostrado. Ao final de cada resultado da busca aparece a opção de reserva ou desistência do quarto. Todas as vezes que um quarto for aceito, o usuário terá que digitar seu CPF e senha para confirmação da reserva. Ao contrário do modelo anterior onde o usuário digitava estes dados uma única vez para reservar qualquer quantidade de quartos. Melhorias podem ser implementadas.

3.6 Implementação utilizando *MySQL, cHTML* e *PHP*

Através do método *POST*, os dados dos formulários são enviados ao servidor e utilizados pelo script *PHP*. As variáveis podem ser utilizadas diretamente pelo script, ou seja, se há um campo denominado Senha no formulário, o *PHP* pode utilizar a variável Senha no código, apenas inserindo o símbolo $ na frente do nome da variável ($senha).

Após o preenchimento de cada formulário do Cadastro de Clientes, os dados são enviados para o servidor. Quando o arquivo CadastroDeClientes5.php envia informações para o servidor, este já está habilitado para realizar o cadastro completo de todos os dados recebidos, com exceção dos dados referentes aos telefones do cliente que já haviam sido cadastrados no momento que o servidor recebeu informações do arquivo CadastroDeClientes4.php.

De forma geral o script *PHP* para cadastro de clientes realiza as seguintes ações:

1. Verifica se o nome, CPF ou Senha são nulos. Se um dos campos for nulo, solicita ao cliente para voltar à página anterior e preencher o campo.

2. Se os campos acima estiverem preenchidos corretamente, o cadastro é efetivado. Para isso, a seguinte instrução *SQL* é utilizada:

```
INSERT INTO cliente
VALUES ('$nome','$rua', '$bairro', '$cidade', '$estado', '$cep', '$pais', $cpf
        '$dtNascimento', '$senha')
```

A comunicação com o banco de dados é realizada através dos comandos:

```
$link = mysql_connect("localhost", "root");
mysql_select_db("ReservasOnLine",$link);
$resultado = mysql_query($insercao, $link);
```

O primeiro comando faz a conexão com o *SGBD*. O segundo seleciona o banco de dados a ser acessado e o terceiro faz com a variável "resultado" receba o resultado da consulta no banco de dados.

3. Para cada telefone preenchido pelo cliente, um cadastro é efetivado com a seguinte instrução *SQL*:

```
INSERT INTO cliente_telefone
VALUES ($cpf,'$telefone1', '$tipo_tel1')
```

4. Se o cliente optar pela inclusão de dependentes em seu cadastro, 6 formulários, um para cada dependente, são apresentados. Caso contrário, uma mensagem de agradecimento aparece.

O arquivo CadastroDeDependentes.php possui duas funções. Funcionando de forma recursiva, ele efetiva o cadastro de um dependente ao mesmo tempo em que, através de um formulário, recolhe dados de um outro dependente. A recursão termina quando o último dependente é cadastrado. Quando o script conecta-se ao banco de dados e insere um dependente a seguinte instrução é usada:

```
INSERT INTO dependente
VALUES ($cpf_cli, '$dependente','$dtNascDep', '$relDep')
```

Quando o cliente realiza uma pesquisa customizada, os dados sobre a cidade, o estado, a categoria do quarto, o tipo de acomodação, o preço e as datas de entrada e de saída são enviados ao servidor. Com estes dados, no arquivo Resultado1.php, uma instrução SQL é montada dinamicamente para que a consulta ao banco de dados possa ser efetivada. A instrução SQL completa possui todas as condições para que a consulta seja customizada:

```
SELECT hot_cnpj, hot_nome, hot_rua, hot_bairro, hot_cidade, hot_estado,
        hot_CEP, hot_homepage, hot_email, qto_numero, qto_tipo,
        qto_nopessoas, qto_preco, qto_descricao
FROM    hotel, quarto
WHERE   hot_cnpj = qto_hot_cnpj AND
        hot_cidade = 'cidade_escolhida' AND
        qto_tipo = 'tipo_escolhido' AND
        qto_nopessoas = número_de_pessoas_escolhido AND
```

```
qto_preco < preco_escolhido AND
(qto_hot_cnpj, qto_numero)

NOT IN(
SELECT res_hot_cnpj, res_noquarto
FROM   reserva
WHERE  (('data_de_entrada' < res_entrada AND
       'data_de_saída' > res_saida)  OR
       (‘data_de_entrada' >= res_entrada  AND
       'data_de_saída' <= res_saida)  OR
       (‘data_de_entrada' < res_entrada  AND
       (' data_de_saída' > res_entrada  AND
       'data_de_saída' < res_saida))  OR
       ((‘data_de_entrada' > res_entrada  AND
       'data_de_entrada' < res_saida)  AND
       'data_de_saída' > res_saida))   ).
```

Depois de realizada a consulta customizada, o arquivo Resultado2.php tratará de mostrar para o cliente informações a respeito dos quartos encontrados. Estas informações incluem: nome e telefone do hotel, tipo de quarto, quantidade de pessoas que poderão se hospedar no quarto, preço da diária e alguma informação complementar a respeito do quarto.

Escolhido o quarto e o hotel, o cliente deve confirmar a reserva preenchendo os campos CPF e a senha, ambos já cadastrados. Isto é feito no arquivo Resultado3.php e os dados são enviados para o arquivo ConfirmaReserva.php, onde a senha é validada e a reserva é efetivada através da instrução

```
INSERT INTO reserva
VALUES ($cpf, $cnpjhotel, $noquarto, '$dtEntrada', '$dtSaida')
```

Obs: Como o *MySQL* não aceita um comando *SELECT* dentro da condição de outra consulta, a verificação dos quartos com reserva foi implementada no *script*. Isto pode afetar o desempenho da arquitetura. Os códigos completos podem ser encontrados no anexo B.

3.7 Implementação utilizando *MySQL, WML* e *PHP*

A implementação em *WML* segue o mesmo padrão definido para *cHTML*. A parte inicial que inclui a configuração do servidor *Apache* para *WML* e criação do banco de dados já foi explicada.

A implementação do código não foi possível de ser completada ficando como proposta para trabalhos futuros.

Capítulo 4

Resultados e Discussão

Este capítulo apresenta uma análise comparativa das duas tecnologias de telefonia celular abordadas até aqui. São elas o *WAP* e o *i-Mode*. Na primeira etapa da comparação, um estudo sobre estas tecnologias foi elaborado para que um levantamento pudesse ser feito visando expor as qualidades e os defeitos de uma tecnologia com relação à outra. Na segunda etapa, a etapa das implementações, o enfoque do estudo foi direcionado as dificuldades que as implementações causariam quando o modelo proposto por [Kin01] fosse modificado para atender às novas necessidades. Ambas as etapas serão discutidas a seguir.

4.1 Diferenças entre *WAP* e *i-Mode*

A Tabela 4.1 apresenta resumidamente as principais diferenças entre *WAP* e *i-Mode*. Observe que *i-Mode* é o nome do serviço oferecido pela *NTT DoCoMo* e, de uma forma distinta, *WAP* é o nome do padrão (protocolo) sobre o qual podem-se construir serviços para serem oferecidos por redes de comunicação móvel. O *i-Mode* tem se popularizado rapidamente desde seu lançamento em fevereiro de 1999 e está cada dia mais acessível aos usuários de telefonia móvel do mundo inteiro. Já o concorrente *WAP*, apesar do sucesso atual, passou por momentos de instabilidades e crises, e ainda sim, com os novos avanços da tecnologia, está novamente crescendo e voltou a conquistar novos usuários.

O serviço *i-Mode* foi desenvolvido sobre um serviço de acesso à Internet, para disponibilizar informações aos usuários do sistema de transmissão por pacotes *PDC-P "personal digital cellular – packet"*. Para que o *i-Mode* fosse desenvolvido e atingisse tamanho sucesso, a *NTT DoCoMo* priorizou o uso de tecnologias que já existiam e que eram padrão para a Internet convencional,

ou seja, linguagem *HTML*, o protocolo *TCP/IP*, etc.

Diferenças entre *WAP* e *i-Mode*		
Parâmetros de comparação	***WAP***	***i-Mode***
O que é?	Protocolo de comunicação usado por dispositivos moveis.	Nome do serviço oferecido pela *NTT DoCoMo*.
Trafego de dados	circuito de troca de dados (*CSD*)	sistema de transmissão por pacotes (*PDC-P*)
Linguagem utilizada	*WML*	*cHTML*
Adaptação de uma página *HTML* para *WML* ou *cHTML*	complexo	simples
Conexão	Descontínua (não é possível fazer uma chamada e navegar-se pela Internet sem que uma delas seja interrompida)	Contínua (permite fazer chamadas e navegar na Internet simultaneamente)
Suportes oferecidos	*JPEG*, tabelas, múltiplos estilos e fontes	*GIF*
Pagamento do serviço	tempo de conecção com a central telefônica	volume de pacotes enviados e recebidos

Tabela 4.1 – Diferenças entre *WAP* e *i-Mode*

Ambas as tecnologias possuem algumas características em comum, mas há também grandes diferenças entre elas, o que traz vantagem de uma em relação à outra. Entretanto, ao se fazer um acompanhamento mais detalhado dos serviços oferecidos tanto pelo *WAP* quanto o *i-Mode*, notamos que o segundo está se adaptando melhor às novas tecnologias que estão surgindo e desta forma está crescendo muito rapidamente. De acordo com especialistas da área, a presença do *i-Mode* em todos os lugares do mundo onde existe o serviço *WAP*, num futuro próximo, é indiscutível. Talvez não chegue a substituir o *WAP*, mas seu uso será bastante disseminado.

4.2 Características da nova implementação

É importante mostrar o quanto algumas aplicações *Web* ficariam inviáveis para um aparelho celular, ou qualquer outro dispositivo, que possua algumas restrições que um computador de mesa não possui. Para se ter uma idéia desta inviabilidade, tomemos como exemplo o item "Cadastro de Clientes". Realizar um cadastro razoavelmente extenso através de um computador de mesa, ou um *Notebook*, é uma tarefa aparentemente fácil, mas o mesmo cadastro feito através de um telefone celular, demanda tempo e paciência. Isso porque os dispositivos móveis, como já foi dito, possuem diferentes tipos de entrada de dados como, por exemplo, a ausência de um teclado completo, como nos *PCs*, o que dificulta bastante o trabalho em digitar textos como: Nome, Endereço, Cidade etc. Portanto a primeira conclusão, a mais óbvia, diz respeito à ineficiência de se fazer cadastros longos através do telefone celular. É sabido que estes aparelhos devem ser usados somente para aplicações simples como escolha de um item em menus, digitação simplificada e navegação entre *cards*. Qualquer outra tarefa que desgaste o usuário causará nele irritação e desinteresse em utilizar o sistema.

A segunda observação diz respeito à forma que se transformou a nova interface de acordo com as exigências do hardware dos celulares. Qualquer página maior deveria ser dividida em pequenos *cards* de tamanhos suficientes para atender o limite dos *buffers* de entrada (512 bytes) e saída (4096 bytes) dos aparelhos. Assim, houve uma grande necessidade de diminuir os recursos da interface ocasionando na redução das informações que eram mostradas na tela do celular. Os resultados que indicavam duas ou mais vagas para um mesmo hotel, não seriam mais mostrados numa única tela, e sim em telas separadas. Esta nova forma de exibição trouxe dificuldades para o cliente que agora não tem como comparar os quartos de um mesmo hotel antes de fazer as reservas. Outra desvantagem diz respeito à maneira como o usuário preenche seu cadastro. No modelo anterior, ele podia ler todos os campos do formulário e corrigir quaisquer erros antes de enviá-lo. Além disso, ele também podia escolher entre as opções ao invés de digitar. A escolha de uma opção num *menu* reduz a possibilidade de erros de digitação como: datas, nome da cidade, do estado etc.

A nova interface trouxe ainda algumas vantagens como simplificação no código e redução no tamanho dos arquivos apesar de seu aumento em quantidade.

Quanto ao código formado por *scripts PHP*, nenhuma alteração considerável teve que ser realizada, mantendo-se quase todo o formato do modelo original. A função *ECHO*, usada para enviar variáveis, foi utilizada inúmeras vezes. Seu uso é indispensável para que informações do cadastro dos clientes e da busca de hotéis sejam enviadas para outros arquivos, até que, de posse de todos as informações necessárias, o servidor possa realizar um cadastro

ou gerar um resultado.

Até aqui os maiores problemas encontrados dizem respeito a realização dos cadastros e da exibição do resultado da pesquisa. O formulário para Pesquisa de Hotéis, no modelo anterior, não continha tantos campos para digitação. Por isso a nova interface ficou quase toda formada por menus de opções. Esta parte do sistema ficou bem ao estilo de um sistema para celulares.

Uma sugestão para continuidade deste trabalho, então, é a seguinte: o Sistema de Reservas em Hotéis poderia ser implementado fazendo o uso de duas plataformas. A primeira seria o cadastro de clientes e dependentes por meio de um *PC*, ou seja, via *Web*. A segunda seria a realização da busca de hotéis por meio de dispositivos móveis, como os celulares. Assim, os clientes fariam seus cadastros no sistema via *Web* (atitude realizada uma única vez) e, daí por diante, estariam aptos a solicitar buscas e reservas de hotéis por meio do seu celular. Esta sugestão torna a aplicação um pouco mais compatível com as exigências e regras impostas pelas plataformas dos dispositivos estudados aqui.

Capítulo 5

Conclusões e Propostas Futuras

O estudo comparativo das tecnologias *i-Mode* e *WML/WAP*, através da implementação de um Sistema de Reservas em Hotéis, contribui com importantes conclusões. Ambas as tecnologias mostraram-se capazes de executar aplicações para *Web*, apesar de alguns inconvenientes como os apresentados na seção 4.2. O desenvolvimento de aplicações para dispositivos móveis, como já era de se esperar, é o grande desafio com o qual se deparam os desenvolvedores da atualidade. O modelo proposto neste trabalho, desenvolvido a partir do modelo já existente, criado por [Kin01], mostra como diversos problemas devem ser tratados para se obter resultados satisfatórios. De forma geral, a maioria dos serviços oferecidos por *sites* na *Web* deve passar cuidadosamente por várias análises a fim de que a usabilidade e a confiabilidade possam ser garantidas antes de serem implementadas as aplicações para as redes *Wireless*.

O modelo proposto neste trabalho foi testado em simuladores. Uma proposta de melhoria seria testá-lo em um dispositivo sem fio real. Além disso, vários parâmetros de comparação poderiam ser mais bem elaborados e, questões do tipo desempenho das aplicações (abordando o tempo de resposta do sistema), portabilidade e segurança podem ainda ser questionadas de várias formas.

Quanto a arquitetura dos dois modelos vistos nesta monografia, trabalhos futuros podem conduzir a melhorias e até chegar a alguma proposta de mudança. E a implementação usando *WML* ainda pode ser feita.

Referências Bibliográficas

[Eurotec] Eurotechnology Japan K. K., 2002. **The Unofficial Independent I-Mode FAQ**. URL: http://www.eurotechnology.com/imode/faq gen.html. Pesquisado em 20/10/2002.

[Fig00] FIGUEIREDO, G. **França está no topo do ranking de turistas estrangeiros**, 2000. Jornal da tarde - Estadao.com.br.URL: http://www.jt.estadao.com.br/suplementos/turi/2000/12/31/ turi021.html. Pesquisado em 18/10/2002.

[Fórum] Wireless Application Protocol Fórum® Ltd, 2000. **Application Protocol Forum Ltd.** URL: http://www.wapforum.org. Pesquisado em 17/10/2002.

[Jup01] Jupiter Media Metrix, 2001. URL: http://www.jmm.com. Pesquisado em 18/10/2002.

[Kam98] KAMADA, T. **Compact HTML for Small Information Appliances**, 1998. W3C® World Wide Web Consortium. URL: http://www.w3.org/TR/1998/NOTEcompactHTML-19980209. Pesquisado em 20/10/2002.

[Kin01] KINOSHITA, V. G. **Banco de Dados Via WEB: Uma Análise Comparativa**, 2001. Monografia (Graduação em Ciência da Computação) - Universidade Federal de Lavras, Lavras/MG.

[Les01] LESSA, R. B. **Estudo sobre a Internet Móvel e o *M-Commerce***, 2001. Monografia (Graduação em Ciência da Computação) – Faculdade de Informática da Universidade Luterana do Brasil, Gravataí/RS.

[Mar99] MARIANO A. **Wireless Networks O padrão IEEE 802.11b para redes sem fio**, 1999. URL: http://www.networkdesigners.com.br/Artigos/wireless/wireless.html.Pesquisado em 11/11/2002.

[Palow] Palowireless - i-mode Browsers and Emulators. URL: http://www.palowireless.com/imode/browsers.asp. Pesquisado em 27/10/2002.

[Ros01] ROSA, H. F. **A Maravilhosa Segunda Geração dos Celulares Japoneses. E a nossa?**, 2001. URL: http://sites.uol.com.br/wirelessbr/docomo.html. Pesquisado em 25/10/2002.

[Ros00] ROSA, H. F. **O "WAP WORLD" - Uma visão geral da "Wireless WEB"**, 2000. URL: http://sites.uol.com.br/helyowap/ccintro2.html. Pesquisado em 25/10/2002.

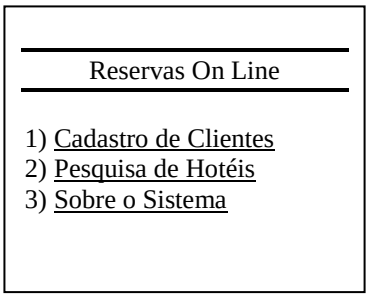
[Yu01] YURI, F. **O Fenômeno I-Mode**, 2001. URL: http://sites.uol.com.br/wirelessbr/imodenoticias.html. Pesquisado em 14/10/2002.

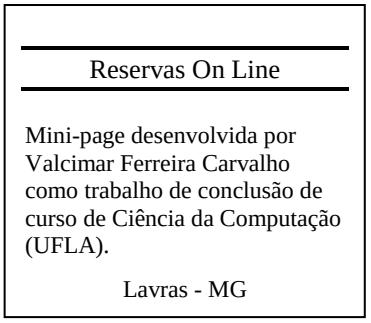
[Spo00] SPOSITO, G. **Desenvolvimento de Aplicações WAP – Arquitetura WAP**, 2000. URL: http://sites.uol.com.br/helyr/sposito/wap_sposito01.html. Pesquisado em 23/07/2002.

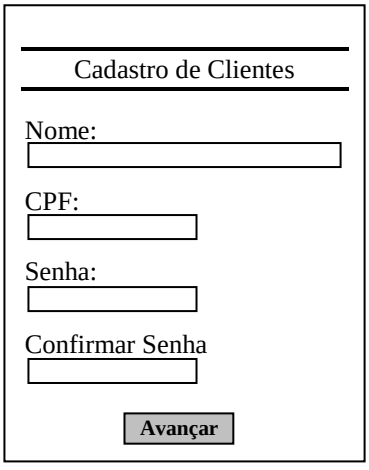
[WMLClub] WMLClub. URL: http://www.br.wmlclub.com/programas/emuladores.htm. Pesquisado em 27/10/2002.

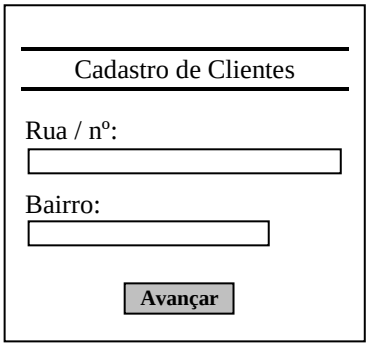
Anexo A

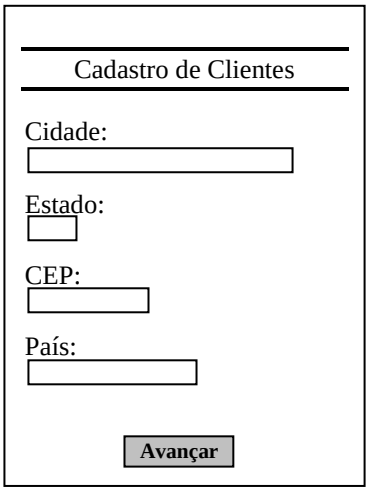
Abaixo estão as *cards* desenvolvidas para o Sistema de Reservas em Hotéis para celulares:

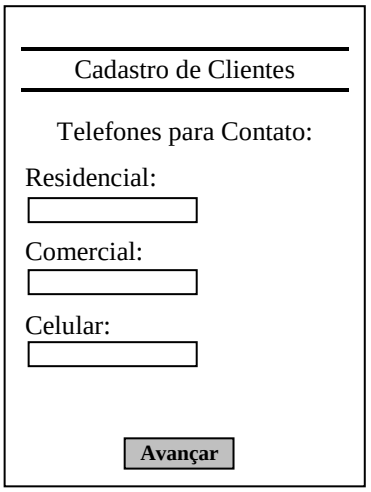


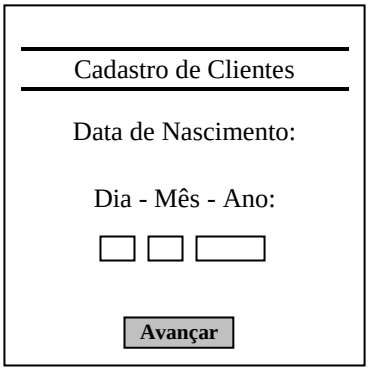


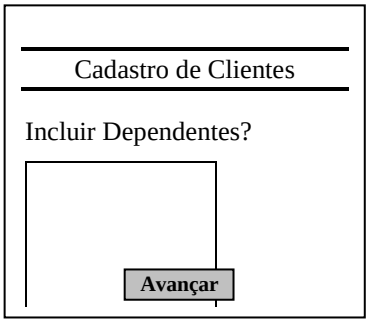


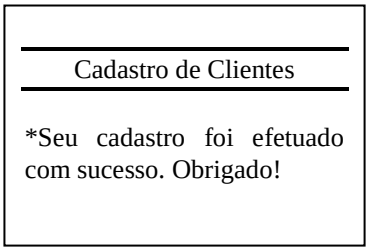


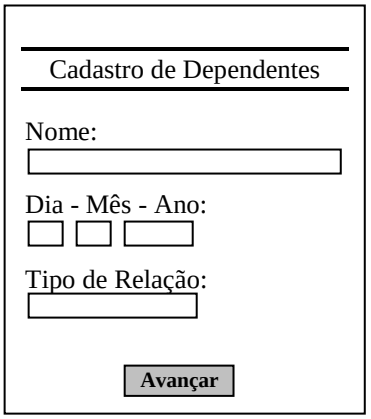

Cadastro de Clientes
Cidade:
Estado:
CEP:
País:
Avançar

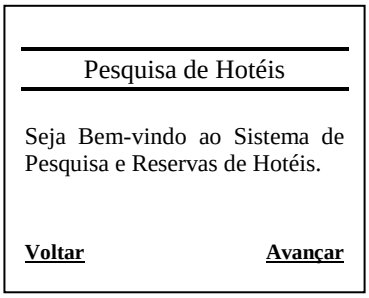

Cadastro de Clientes
Telefones para Contato:
Residencial:
Comercial:
Celular:
Avançar


Cadastro de Clientes
Data de Nascimento:
Dia - Mês - Ano:
Avançar

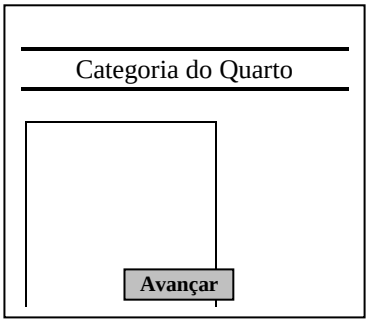

Cadastro de Clientes
Incluir Dependentes?
Avançar

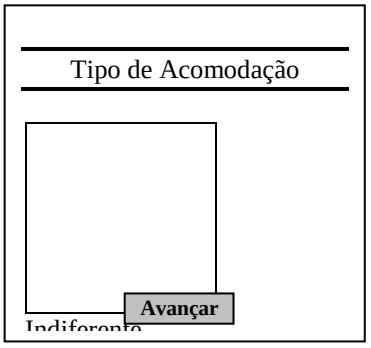

Cadastro de Clientes
*Seu cadastro foi efetuado com sucesso. Obrigado!

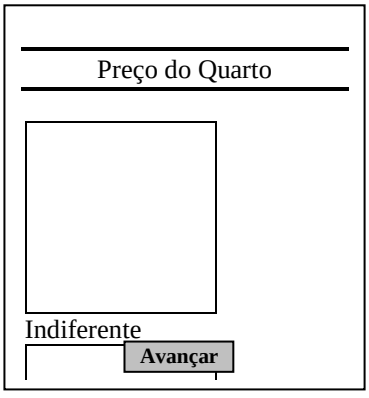

Cadastro de Dependentes
Nome:
Dia - Mês - Ano:
Tipo de Relação:
Avançar

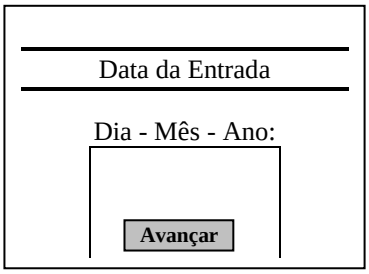

Pesquisa de Hotéis
Seja Bem-vindo ao Sistema de Pesquisa e Reservas de Hotéis.
Voltar
Avançar

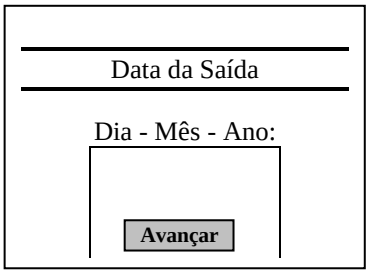

Pesquisa de Hotéis
Informe a Localidade:
Cidade:
Estado:
Avançar

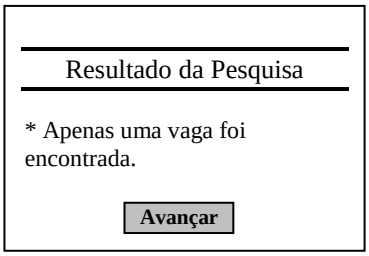

Categoria do Quarto
Avançar

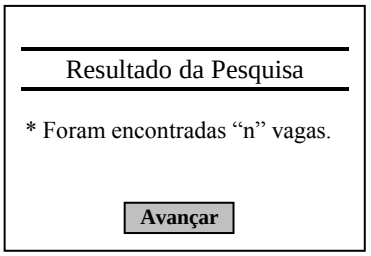

Tipo de Acomodação
Avançar

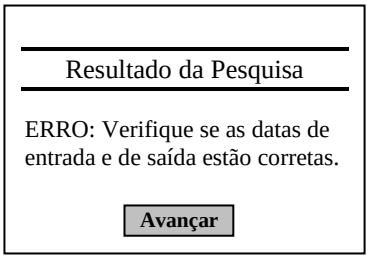

Preço do Quarto
Indiferente
Avançar

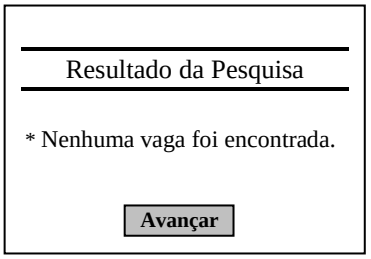

Data da Entrada
Dia - Mês - Ano:
Avançar

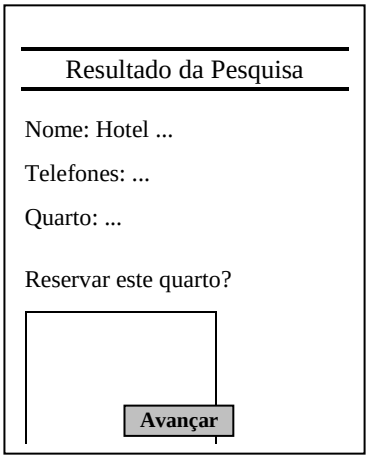

Data da Saída
Dia - Mês - Ano:
Avançar

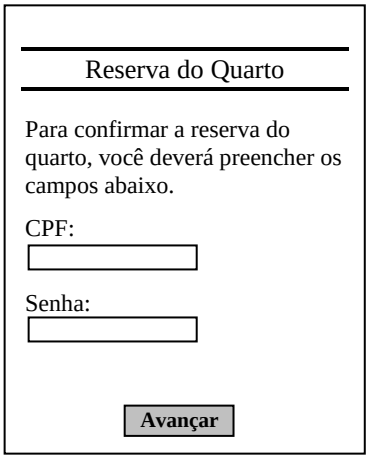

Resultado da Pesquisa
* Apenas uma vaga foi encontrada.
Avançar

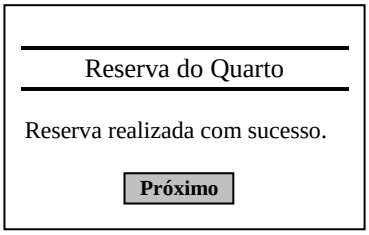

Resultado da Pesquisa
* Foram encontradas “n” vagas.
Avançar

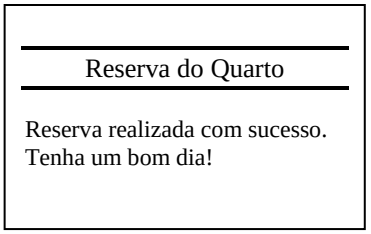

Resultado da Pesquisa
ERRO: Verifique se as datas de entrada e de saída estão corretas.
Avançar

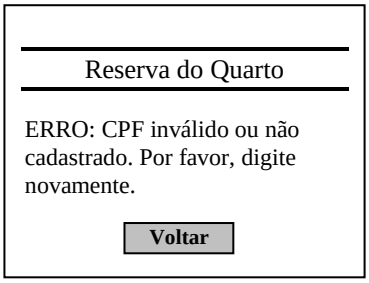

Resultado da Pesquisa
* Nenhuma vaga foi encontrada.
Avançar

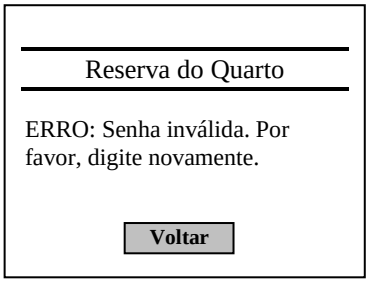

Resultado da Pesquisa
Nome: Hotel ...
Telefones: ...
Quarto: ...
Reservar este quarto?
Avançar


Reserva do Quarto
Para confirmar a reserva do quarto, você deverá preencher os campos abaixo.
CPF:
Senha:
Avançar


Reserva do Quarto
Reserva realizada com sucesso.
Próximo


Reserva do Quarto
Reserva realizada com sucesso.
Tenha um bom dia!


Reserva do Quarto
ERRO: CPF inválido ou não cadastrado. Por favor, digite novamente.
Voltar


Reserva do Quarto
ERRO: Senha inválida. Por favor, digite novamente.
Voltar

Anexo B

1 Arquivo index.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0 Draft//EN">
<HTML>
   <HEAD><TITLE>Reservas On Line</TITLE></HEAD>
   <BODY>
      <div align="center"><font size="4">
      <HR><b>Reservas On Line</b></font></div><HR>
      1)<A HREF="CadastroDeClientes1.php" accesskey="1">
      Cadastro de Clientes<BR>
      </A>
      2)<A HREF="PesquisaDeHoteis.htm" accesskey="2">
      Pesquisa de Hoteis<BR>
      </A>
      3)<A HREF="Sobre.HTM" accesskey="3">
      Sobre o Sistema<BR>
      </A>
   </BODY>
</HTML>
```

2 Arquivo Sobre.htm

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0 Draft//EN">
<HTML>
   <HEAD><TITLE>Reservas On Line</TITLE></HEAD>
   <BODY>
      <div align="center"><font size="4">
      <HR><b>Reservas On Line</b></font></div><HR>
      <font size="2">
      Mini-page desenvolvida por Valcimar Ferreira Carvalho como
      trabalho de conclusão do curso de Ciência da Computação
      (UFLA) - Lavras - MG
      </font>
   </BODY>
```

```
</HTML>
```

3 Arquivo CadastroDeClientes1.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0 Draft//EN">
<HTML>
   <HEAD><TITLE>Cadastro de Clientes</TITLE></HEAD>
   <BODY>
      <div align="center"><font size="4">
      <HR><b>Cadastro de Clientes</b></font></div><HR>
      <FORM method="POST" action="CadastroDeClientes2.php">
      <font size="2">
      Nome:
      <div><td width="56%" height="24"><input type="text"
      name="nome" size="25"></td></div>
      <br>CPF:
      <div><td width="56%" height="21"><input type="text"
      name="cpf" size="12"></td></div>
      <br>Senha:
      <div><td width="56%" height="21"><input type="password"
      name="senha" size="12"></td></tr></div>
      <br>Confirmar Senha:
      <div><td width="56%" height="21"><input type="password"
      name="senha2" size="12"></td></div><p>
      </font>
      <p align="center"><b><input type="submit" value="Avançar"
      name="B1"></b></p>
      </FORM>
   </BODY>
</HTML>
```

4 Arquivo CadastroDeClientes2.php

```
<? PHP
if ($nome == '')  // O campo nome nao foi preenchido
{?>
   <!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0
   Draft//EN">
   <HTML>
      <HEAD><TITLE>Cadastro de Clientes</TITLE></HEAD>
      <BODY>
         <div align="center"><font size="4">
         <HR><b>Cadastro de Clientes</b></font></div><HR>
         <p>ERRO: O nome não foi preenchido. Retorne e preencha o
         campo.</p>
         (1)<A HREF="CadastroDeClientes1.php"
         accesskey="1">Voltar</A>
      </BODY>
   </HTML>
```

```
<?}

else if ($cpf == '')  //cpf nao foi preenchido
{?>
   <!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0
   Draft//EN">
   <HTML>
      <HEAD><TITLE>Cadastro de Clientes</TITLE></HEAD>
      <BODY>
         <div align="center"><font size="4">
         <HR><b>Cadastro de Clientes</b></font></div><HR>
         <p>ERRO: O CPF nao foi preenchido. Retorne e preencha o
         campo.</p>
         (1)<A HREF="CadastroDeClientes1.php"
         accesskey="1">Voltar</A>
      </BODY>
   </HTML>
<?}

//senha nao foi preenchida ou foi confirmada incorretamente
else if (($senha == '') or ($senha != $senha2))
{?>
   <!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0
   Draft//EN">
   <HTML>
      <HEAD><TITLE>Cadastro de Clientes</TITLE></HEAD>
      <BODY>
         <div align="center"><font size="4">
         <HR><b>Cadastro de Clientes</b></font></div><HR>
         <p>ERRO: A senha nao foi preenchida ou foi confirmada
         incorretamente. Retorne e preencha os campos
         novamente.</p>
         (1)<A HREF="CadastroDeClientes1.php"
         accesskey="1">Voltar</A>
      </BODY>
   </HTML>
<?}

else //inclusao do nome, cpf e senha do cliente
{?>
   <!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0
   Draft//EN">
   <HTML>
      <HEAD><TITLE>Cadastro de Clientes</TITLE></HEAD>
      <BODY>
         <div align="center"><font size="4">
         <HR><b>Cadastro de Clientes</b></font></div><HR>
         <FORM method="POST" action="CadastroDeClientes3.php">
         <font size="2">
```

```
         <br>Rua / n°:
         <div><td width="56%" height="25"><input type="text"
         name="rua" size="30"></td></div>
         <br>Bairro:
         <div><td width="56%" height="21"><input type="text"
         name="bairro" size="20"></td></div>
         </font>
         <?
         //Enviando dados para se realizar o cadastro completo
         mais adiante.
         echo "<input type='hidden' name='nome' value='$nome'>";
         echo "<input type='hidden' name='cpf' value='$cpf'>";
         echo "<input type='hidden' name='senha' value='$senha'>";
         ?>
         <p align="center"><b><input type="submit" value="Avançar"
         name="B1"></b></p>
         </FORM>
      </BODY>
   </HTML>
<?}
```

5 Arquivo CadastroDeClientes3.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0 Draft//EN">
<HTML>
   <HEAD><TITLE>Cadastro de Clientes</TITLE></HEAD>
   <BODY>
      <div align="center"><font size="4">
      <HR><b>Cadastro de Clientes</b></font></div><HR>
      <FORM method="POST" action="CadastroDeClientes4.php">
      <font size="2">
      <br>Cidade:
      <div><td width="56%" height="21"><input type="text"
      name="cidade" size="25"></td></div>
      <br>Estado:
      <div><td width="56%" height="24"><input type="text"
      name="estado" size="2"></td></div>
      <br>CEP:
      <div><td width="56%" height="21"><input type="text"
      name="cep" size="9"></td></div>
      <br>País:
      <div><td width="56%" height="21"><input type="text"
      name="pais" size="12"></td></div>
      </font>
      <?
      echo "<input type='hidden' name='nome' value='$nome'>";
      echo "<input type='hidden' name='cpf' value='$cpf'>";
      echo "<input type='hidden' name='senha' value='$senha'>";
```

```
      echo "<input type='hidden' name='rua' value='$rua'>";
      echo "<input type='hidden' name='bairro' value='$bairro'>";
      ?>
      <p align="center"><b><input type="submit" value="Avançar"
      name="B1"></b></p>
      </FORM>
   </BODY>
</HTML>
```

6 Arquivo CadastroDeClientes4.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0 Draft//EN">
<HTML>
   <HEAD><TITLE>Cadastro de Clientes</TITLE></HEAD>
   <BODY>
      <div align="center"><font size="4">
      <HR><b>Cadastro de Clientes</b></font></div><HR>
      <FORM method="POST" action="CadastroDeClientes5.php">
      <p align="center"><b>Telefones para Contato:</b></p>

      <font size="2">
      Residencial:
      <div><td width="56%" height="21"><input type="text"
      name="telefone1" size="9"></td></div>
      <br>Comercial:
      <div><td width="56%" height="21"><input type="text"
      name="telefone2" size="9"></td></div>
      <br>Celular:
      <div><td width="56%" height="21"><input type="text"
      name="telefone3" size="10"></td></div>
      </font>
      <?
      echo "<input type='hidden' name='nome' value='$nome'>";
      echo "<input type='hidden' name='cpf' value='$cpf'>";
      echo "<input type='hidden' name='senha' value='$senha'>";
      echo "<input type='hidden' name='rua' value='$rua'>";
      echo "<input type='hidden' name='bairro' value='$bairro'>";
      echo "<input type='hidden' name='cidade' value='$cidade'>";
      echo "<input type='hidden' name='estado' value='$estado'>";
      echo "<input type='hidden' name='cep' value='$cep'>";
      echo "<input type='hidden' name='pais' value='$pais'>";
      ?>
      <p align="center"><b><input type="submit" value="Avançar"
      name="B1"></b></p>
      </FORM>
   </BODY>
</HTML>
```

7 Arquivo CadastroDeClientes5.php

```
<?PHP
$link = mysql_connect("localhost", "root") or die("Não foi
possivel conectar o BD");
mysql_select_db("ReservasOnLine",$link);
if($telefone1 != '')
{
   $tipo_tel1 = "residencial";
   $insercaotel1 = "INSERT INTO cliente_telefone VALUES
   ('$cpf','$telefone1', '$tipo_tel1')";
   $resultado = mysql_query($insercaotel1, $link);
   if (!$resultado)
   {
      echo("ERRO: " . mysql_error() . "\n$insercao\n");
      exit();
   }
}
if($telefone2 != '')
{
   $tipo_tel2 = "comercial";
   $insercaotel2 = "INSERT INTO cliente_telefone VALUES
   ('$cpf','$telefone2', '$tipo_tel2')";
   $resultado = mysql_query($insercaotel2, $link);
   if (!$resultado)
   {
      echo("ERRO: " . mysql_error() . "\n$insercao\n");
      exit();
   }
}
if($telefone3 != '')
{
   $tipo_tel3 = "celular";
   $insercaotel3 = "INSERT INTO cliente_telefone VALUES
   ('$cpf','$telefone3', '$tipo_tel3')";
   $resultado = mysql_query($insercaotel3, $link);
   if (!$resultado)
   {
      echo("ERRO: " . mysql_error() . "\n$insercao\n");
      exit();
   }
}
mysql_close($link);
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0 Draft//EN">
<HTML>
   <HEAD><TITLE>Cadastro de Clientes</TITLE></HEAD>
   <BODY>
```

```
      <div align="center"><font size="4">
      <HR><b>Cadastro de Clientes</b></font></div><HR>
      <FORM method="POST" action="CadastroDeClientes6.php">
      <font size="2">
      <p align="center"><b>Data de Nascimento</b>
      <div><br><p align="center"> Dia - Mês - Ano : </p>
         <p align="center">
         <td width="56%" height="24"><input type="text"
         name="diaNasc" size="2"></td> -
         <td width="56%" height="24"><input type="text"
         name="mesNasc" size="2"></td> -
         <td width="56%" height="24"><input type="text"
         name="anoNasc" size="4"></td>
         </p>
      </div>
      <?
      echo "<input type='hidden' name='nome' value='$nome'>";
      echo "<input type='hidden' name='cpf' value='$cpf'>";
      echo "<input type='hidden' name='senha' value='$senha'>";
      echo "<input type='hidden' name='rua' value='$rua'>";
      echo "<input type='hidden' name='bairro' value='$bairro'>";
      echo "<input type='hidden' name='cidade' value='$cidade'>";
      echo "<input type='hidden' name='estado' value='$estado'>";
      echo "<input type='hidden' name='cep' value='$cep'>";
      echo "<input type='hidden' name='pais' value='$pais'>";
      ?>
      </font>
      <p align="center"><b><input type="submit" value="Avançar"
      name="B1"></b></p>
      </FORM>
   </BODY>
</HTML>
```

8 Arquivo CadastroDeClientes6.php

```
<? PHP
//Cadastrando todos os dados do cliente
$dtnascimento = "$anoNasc";
$dtnascimento .= "-$mesNasc";
$dtnascimento .= "-$diaNasc";
$link = mysql_connect("localhost", "root") or die("Não foi
possivel conectar o BD");
mysql_select_db("ReservasOnLine",$link);
$insercao = "INSERT INTO cliente VALUES
('$nome','$rua', '$bairro', '$cidade', '$estado', '$cep',
'$pais', '$cpf','$dtnascimento', '$senha')";
$resultado = mysql_query($insercao, $link);
if (!$resultado)
```

```
{
   echo("ERRO: " . mysql_error() . "\n$insercao\n");
   exit();
}
mysql_close($link);
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0 Draft//EN">
<HTML>
   <HEAD><TITLE>Cadastro de Clientes</TITLE></HEAD>
   <BODY>
      <div align="center"><font size="4">
      <HR><b>Cadastro de Clientes</b></font></div><HR>
      <FORM method="POST" action="CadastroDeDependentes.php">
      <font size="2">
      <p>Incluir Dependentes?</p>
      <tr><td>
      <input type="radio" value="V1" name="inclusao">Sim </p>
      <input type="radio" value="V2" checked name="inclusao">
      Não</p></td></tr>
      </font>
      <?
      $num_dependente = 1; // Variavel para controlar o cadastro
      de dependentes
      echo"<input type='hidden' name='cpf_cli' value='$cpf'>";
      echo"<input type='hidden' name='num_dependente'
      value='$num_dependente'>";
      ?>
      <p align="center"><b><input type="submit" value="Avançar"
      name="B1"></b></p>
      </FORM>
   </BODY>
</HTML>
```

9 Arquivo CadastroDeDependentes.php

```
<?PHP
if ($inclusao == V2) //nao vai cadastrar dependentes
{?>
   <!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0
   Draft//EN">
   <HTML>
      <HEAD><TITLE>Cadastro de Clientes</TITLE></HEAD>
      <BODY>
         <div align="center"><font size="4">
         <HR><b>Cadastro de Clientes</b></font></div><HR>
         *Seu cadastro foi efetuado com sucesso. Obrigado!<p>
         (1)<A HREF="index.html" accesskey="1">Voltar</A>
      </BODY>
```

```
   </HTML>
<?}

else if ($inclusao == V1)  // vai cadastrar o(s) dependente(s)
{
   if  ($num_dependente  ==  7)  //  Limite  para  o  cadastro  de
dependentes
   {
      //inclusão do ultimo dependente
      $dtNascDep = "$anoNascDep";
      $dtNascDep .= "-$mesNascDep";
      $dtNascDep .= "-$diaNascDep";
      $link = mysql_connect("localhost", "root");
      mysql_select_db("ReservasOnLine",$link);
      $insercao_dep = "INSERT INTO dependente VALUES
      ('$cpf_cli', '$dependente','$dtNascDep', '$relDep')";
      $resultado = mysql_query($insercao_dep, $link);
      if (!$resultado)
      {
         echo("ERRO: " . mysql_error() . "\n$insercao\n");
         exit();
      }
      mysql_close($link);
      ?>
      <!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0
      Draft//EN">
      <HTML>
         <HEAD><TITLE>Cadastro de Clientes</TITLE></HEAD>
         <BODY>
            <div align="center"><font size="4">
            <HR><b>Cadastro de Clientes</b></font></div><HR>
            *Seu cadastro foi efetuado com sucesso. Obrigado!<p>
            (1)<A HREF="index.html" accesskey="1">Voltar</A>
         </BODY>
      </HTML>
   <?}

   else
   {?>
      <!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0
      Draft//EN">
      <HTML>
         <HEAD><TITLE>Cadastro de Dependentes</TITLE></HEAD>
         <BODY>
            <div align="center"><font size="4">
            <HR><b>Cadastro de Dependentes (<?echo
            $num_dependente;?>)</b></font></div><HR>
            <FORM method="POST"
            action="CadastroDeDependentes.php">
```

```
<?
if ($num_dependente > 1)
{
   //inclusão do dependente do formulario anterior
   $dtNascDep = "$anoNascDep";
   $dtNascDep .= "-$mesNascDep";
   $dtNascDep .= "-$diaNascDep";
   $link = mysql_connect("localhost", "root");
   mysql_select_db("ReservasOnLine",$link);
   $insercao_dep = "INSERT INTO dependente VALUES
   ('$cpf_cli', '$dependente','$dtNascDep',
   '$relDep')";
   $num_dependente++;
   $resultado = mysql_query($insercao_dep, $link);
   if (!$resultado)
   {
      echo("ERRO: " . mysql_error() .
      "\n$insercao\n");
      exit();
   }
   mysql_close($link);
}?>
<font size="2">
<br>Nome:
<div><td width="56%" height="24"><input type="text"
name="dependente" size="25"></td></div>
<br>Nascimento (D/M/A):
<div><td width="56%" height="24"><input type="text"
name="diaNascDep" size="2"></td> -
<td width="56%" height="24"><input type="text"
name="mesNascDep" size="2"></td> -
<td width="56%" height="24"><input type="text"
name="anoNascDep" size="4"></td>
</div><br>Tipo de Relação:
<div><td width="56%" height="24"><input type="text"
name="relDep" size="15"></td></div>
</font>
<?
if ($num_dependente == 1)
   $num_dependente++;
echo"<input type='hidden' name='cpf_cli'
value='$cpf_cli'>";
echo"<input type='hidden' name='inclusao'
value='$inclusao'>";
echo"<input type='hidden' name='num_dependente'
value='$num_dependente'>";
?>
<p align="center"><b><input type="submit"
value="Avançar"
```

```
            name="B1"></b></p>
            </FORM>
         </BODY>
      </HTML>
   <?}
}
```

10 Arquivo PesquisaDeHoteis.htm

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0 Draft//EN">
<HTML>
   <HEAD><TITLE>Pesquisa de Hotéis</TITLE></HEAD>
   <BODY>
      <div align="center"><font size="4">
      <HR><b>Pesquisa de Hotéis</b></font></div><HR>
      <p>Seja Bem-vindo ao Sistema de Pesquisa e Reservas de
      Hotéis.</p>
      (1)<A HREF="index.html" accesskey="1">Voltar</A>
      (2)<A HREF="Localidade.php" accesskey="2">Avançar<BR></A>
   </BODY>
</HTML>
```

11 Arquivo Localidade.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0 Draft//EN">
<HTML>
   <HEAD><TITLE>Pesquisa de Hotéis</TITLE></HEAD>
   <BODY>
      <div align="center"><font size="4">
      <HR><b>Pesquisa de Hotéis</b></font></div><HR>
      <FORM method="POST" action="CatQuarto.php">
      <p align="center"><b>Informe a Localidade</b>
      <br><font size="2">Cidade:
      <div><td width="56%" height="24"><input type="text"
      name="cidade" size="12"></td></div></font>
      <br><font size="2">Estado:
      <div><td width="56%" height="24"><input type="text"
      name="estado" size="2"></td></div></font>
      <p align="center"><b><input type="submit" value="Avançar"
      name="B1"></b></p>
      </FORM>
   </BODY>
</HTML>
```

12 Arquivo CatQuarto.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0 Draft//EN">
<HTML>
   <HEAD><TITLE>Pesquisa de Hotéis</TITLE></HEAD>
   <BODY>
      <div align="center"><font size="4">
      <HR><b>Categoria do Quarto</b></font></div><HR>
      <FORM method="POST" action="TipoAcomod.php">
      <tr><td><font size="2">
      <input type="radio" value="Indiferente" checked
      name="tipoQuarto">Indiferente</p>
      <input type="radio" value="Standard"
      name="tipoQuarto">Standard</p>
      <input type="radio" value="Luxo" name="tipoQuarto">Luxo</p>
      <input type="radio" value="Suíte"
      name="tipoQuarto">Suíte</p></td>
      </font></tr>
      <?
      echo "<input type='hidden' name='estado' value='$estado'>";
      echo "<input type='hidden' name='cidade' value='$cidade'>";
      ?>
      <p align="center"><b><input type="submit" value="Avançar"
      name="B1"></b></p>
      </FORM>
   </BODY>
</HTML>
```

13 Arquivo TipoAcomod.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0 Draft//EN">
<HTML>
   <HEAD><TITLE>Pesquisa de Hotéis</TITLE></HEAD>
   <BODY>
      <div align="center"><font size="4">
      <HR><b>Tipo de Acomodação</b></font></div><HR>
      <FORM method="POST" action="PrecoQuarto.php">
      <tr><font size="2"><td>
         <input type="radio" value="V1" checked
         name="qtd">Indiferente</p>
         <input type="radio" value="V2" name="qtd">Solteiro</p>
         <input type="radio" value="V3" name="qtd">Duplo</p>
         <input type="radio" value="V4" name="qtd">Triplo</p>
         <input type="radio" value="V5"
         name="qtd">Quádruplo</p></td>
      </tr></font>
      <?
      echo "<input type='hidden' name='estado' value='$estado'>";
      echo "<input type='hidden' name='cidade' value='$cidade'>";
      echo "<input type='hidden' name='tipoQuarto'
```

```
      value='$tipoQuarto'>";
      ?>
      <p align="center"><b><input type="submit" value="Avançar"
      name="B1"></b></p>
      </FORM>
   </BODY>
</HTML>
```

14 Arquivo PrecoQuarto.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0 Draft//EN">
<HTML>
   <HEAD><TITLE>Pesquisa de Hotéis</TITLE></HEAD>
   <BODY>
      <div align="center"><font size="4">
      <HR><b>Preço do Quarto</b></font></div><HR>
      <FORM method="POST" action="DataEnt.php">
      <tr><td><font size="2">
      <input type="radio" value="V1" checked
      name="preco">Indiferente</p>
      <input type="radio" value="V2" name="preco">Até R$30,00</p>
      <input type="radio" value="V3" name="preco">De R$30,00 até
      R$50,00</p>
      <input type="radio" value="V4" name="preco">De R$50,00 até
      R$70,00</p>
      <input type="radio" value="V5" name="preco">De R$70,00 até
      $100,00</p>
      <input type="radio" value="V6" name="preco">Acima de
      R$100,00</p></font></td>
      </tr>
      <?
      echo "<input type='hidden' name='estado' value='$estado'>";
      echo "<input type='hidden' name='cidade' value='$cidade'>";
      echo "<input type='hidden' name='tipoQuarto'
      value='$tipoQuarto'>";
      echo "<input type='hidden' name='qtd' value='$qtd'>";
      ?>
      <p align="center"><b><input type="submit" value="Avançar"
      name="B1"></b></p>
      </FORM>
   </BODY>
</HTML>
```

15 Arquivo DataEnt.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0 Draft//EN">
<HTML>
```

```
   <HEAD><TITLE>Pesquisa de Hotéis</TITLE></HEAD>
   <BODY>
      <div align="center"><font size="4">
      <HR><b>Data da Entrada</b></font></div><HR>
      <FORM method="POST" action="DataSaida.php">
      <p align="center"><b>Data da Entrada</b>
      <div><br><font size="2"><p align="center"> Dia - Mês - Ano :
      </p> <p align="center">
      <td width="56%" height="24"><input type="text" name="diaEnt"
      size="2"></td> -
      <td width="56%" height="24"><input type="text" name="mesEnt"
      size="2"></td> -
      <td width="56%" height="24"><input type="text" name="anoEnt"
      size="4"></td>
      </p></font></div>
      <?
      echo "<input type='hidden' name='estado' value='$estado'>";
      echo "<input type='hidden' name='cidade' value='$cidade'>";
      echo "<input type='hidden' name='tipoQuarto'
      value='$tipoQuarto'>";
      echo "<input type='hidden' name='qtd' value='$qtd'>";
      echo "<input type='hidden' name='preco' value='$preco'>";
      ?>
      <p align="center"><b><input type="submit" value="Avançar"
      name="B1"></b></p>
      </FORM>
   </BODY>
</HTML>
```

16 Arquivo DataSaida.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0 Draft//EN">
<HTML>
   <HEAD><TITLE>Pesquisa de Hotéis</TITLE></HEAD>
   <BODY>
      <div align="center"><font size="4">
      <HR><b>Data da Saída</b></font></div><HR>
      <FORM method="POST" action="Resultado1.php">
      <p align="center"><b>Data da Saída</b>
      <div><br><font size="2"><p align="center"> Dia - Mês - Ano :
      </p> <p align="center">
      <td width="56%" height="24"><input type="text"
      name="diaSaida" size="2"></td> -
      <td width="56%" height="24"><input type="text"
      name="mesSaida" size="2"></td> -
      <td width="56%" height="24"><input type="text"
      name="anoSaida" size="4"></td>
      </p></font></div>
```

```
      <?
      $dtEntrada = "$anoEnt";
      $dtEntrada .= "$mesEnt";
      $dtEntrada .= "$diaEnt";
      echo "<input type='hidden' name='estado' value='$estado'>";
      echo "<input type='hidden' name='cidade' value='$cidade'>";
      echo "<input type='hidden' name='tipoQuarto'
      value='$tipoQuarto'>";
      echo "<input type='hidden' name='qtd' value='$qtd'>";
      echo "<input type='hidden' name='preco' value='$preco'>";
      echo "<input type='hidden' name='dtEntrada'
      value='$dtEntrada'>";
      ?>
      <p align="center"><b><input type="submit" value="Avançar"
      name="B1"></b></p>
      </FORM>
   </BODY>
</HTML>
```

17 Arquivo Resultado1.php

```
<?PHP
$dtAtual = date("Ymd");
$dtSaida = "$anoSaida";
$dtSaida .= "$mesSaida";
$dtSaida .= "$diaSaida";

// Verifica se as datas estao corretas, ou seja, se entrada eh
// menor do que saida e se ambas sao maiores que a data atual.
if(($dtEntrada >= $dtAtual) and ($dtSaida >= $dtAtual) and
($dtEntrada <= $dtSaida))
{
   $link = mysql_connect("localhost", "root") or die("Não foi
   possivel conectar o BD");
   mysql_select_db("ReservasOnLine",$link);
   // Consulta de todos os hoteis e quartos cadastrados no BD.
   $consulta = "SELECT hot_cnpj, hot_nome, hot_rua, hot_bairro,
   hot_cidade, hot_estado, hot_CEP, hot_homepage, hot_email,
   qto_numero, qto_tipo, qto_nopessoas, qto_preco, qto_descricao
   FROM hotel, quarto
   WHERE hot_cnpj = qto_hot_cnpj ";
   if ($estado != '')
   {
      $consulta .= "AND hot_estado = '$estado' ";
   }
   if ($cidade != '')
   {
      $consulta .= "AND hot_cidade = '$cidade' ";
```

```
}
// Tipo do quarto
if ($tipoQuarto != 'Indiferente')
{
   $consulta .= " AND qto_tipo = '$tipoQuarto' ";
}
// Quantidade de pessoas
if ($qtd != 'V1')
{
   if ($qtd == 'V2')
      $qtd_pessoas = 1;
   else if ($qtd == 'V3')
      $qtd_pessoas = 2;
   else if ($qtd == 'V4')
      $qtd_pessoas = 3;
   else if ($qtd == 'V5')
      $qtd_pessoas = 4;
   $consulta .= "AND qto_nopessoas = '$qtd_pessoas'";
}

// Preço da diaria
if ($preco != 'V1')
{
   if ($preco == 'V2')
      $consulta .= "AND qto_preco < 30 ";
   else if ($preco == 'V3')
      $consulta .= "AND qto_preco >= 30 AND qto_preco < 50 ";
   else if ($preco == 'V4')
      $consulta .= "AND qto_preco >= 50 AND qto_preco < 70 ";
   else if ($preco == 'V5')
      $consulta .= "AND qto_preco >= 70 AND qto_preco < 100 ";
   else if ($preco == 'V6')
      $consulta .= "AND qto_preco >= 100 ";
}

// Datas
$consulta .= "AND (qto_hot_cnpj, qto_numero) NOT IN ";
$consulta .= "(SELECT res_hot_cnpj, res_noquarto
FROM reserva WHERE ( ";
$consulta .= "('$dtEntrada' < res_entrada AND
'$dtSaida' > res_saida) OR ";
$consulta .= "('$dtEntrada' >= res_entrada AND
'$dtSaida' <= res_saida) OR ";
$consulta .= "('$dtEntrada' < res_entrada AND
('$dtSaida' > res_entrada AND '$dtSaida' < res_saida)) OR ";
$consulta .= "(('$dtEntrada' > res_entrada AND
'$dtEntrada' < res_saida) AND '$dtSaida' > res_saida))) ";
$consulta .= " ; ";
$resultado = mysql_query($consulta, $link);
```

```
if (!$resultado)
{
   echo("ERRO: " . mysql_error() . "\n$insercao\n");
   exit();
}

mysql_close($link);
 $num_linhas = mysql_num_rows($resultado);
 if ($num_linhas == 0) // Nenhuma vaga de hotel foi encontrada
{?>
   <!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0
   Draft//EN">
   <HTML>
      <HEAD><TITLE>Pesquisa de Hotéis</TITLE></HEAD>
      <BODY>
         <div align="center"><font size="4">
         <HR><b>Resultado da Pesquisa</b></font></div><HR>
         <p>*Nenhuma vaga foi encontrada.</p>
         (1)<A HREF="index.html" accesskey="1">Voltar</A>
      </BODY>
   </HTML>
<?}
else // Uma ou mais vagas foram encontradas
{?>
   <!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0
   Draft//EN">
   <HTML>
      <HEAD><TITLE>Pesquisa de Hotéis</TITLE></HEAD>
      <BODY>
         <div align="center"><font size="4">
         <HR><b>Resultado da Pesquisa</b></font></div><HR>
         <FORM method="POST" action="Resultado2.php">
         <?
         if ($num_linhas == 1)
            echo("*Apenas uma vaga foi encontrada.");
         else
         {
            echo("*Foram encontradas ");
            echo $num_linhas;
            echo(" vagas.");
         }
         // Enviando o resultado para o arquivo
         // "Resultado2.php"


         while($tupla = mysql_fetch_array($resultado))
         {
            $nomes_hoteis = $tupla["hot_nome"];
            $cnpj_hoteis = $tupla["hot_cnpj"];
```

```
                $qto_numero = $tupla["qto_numero"];
                $qto_tipo = $tupla["qto_tipo"];
                $qto_nopessoas = $tupla["qto_nopessoas"];
                $qto_preco = $tupla["qto_preco"];
                $qto_descricao = $tupla["qto_descricao"];
                echo "<input type='hidden' name='nomes_hoteis[]'
                value='$nomes_hoteis'>";
                echo "<input type='hidden' name='cnpj_hoteis[]'
                value='$cnpj_hoteis'>";
                echo "<input type='hidden' name='qto_numero[]'
                value='$qto_numero'>";
                echo "<input type='hidden' name='qto_tipo[]'
                value='$qto_tipo'>";
                echo "<input type='hidden'
                name='qto_nopessoas[]' value='$qto_nopessoas'>";
                echo "<input type='hidden' name='qto_preco[]'
                value='$qto_preco'>";
                echo "<input type='hidden'
                name='qto_descricao[]' value='$qto_descricao'>";
             }
             $contador = 0;
             echo "<input type='hidden' name='contador'
             value='$contador'>";
             echo "<input type='hidden' name='num_hoteis'
             value='$num_linhas'>";
             ?>
             <p align="center"><b><input type="submit"
             value="Avançar" name="B1"></b></p>
             </FORM>
          </BODY>
       </HTML>
   <?}
} // fim do if de verificacao das datas

else // Erro nas datas de entrada e de saída
{?>
   <!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0
   Draft//EN">
   <HTML>
      <HEAD><TITLE>Pesquisa de Hotéis</TITLE></HEAD>
      <BODY>
         <div align="center"><font size="4">
         <HR><b>Resultado da Pesquisa</b></font></div><HR>
         <FORM method="POST" action="DataEnt.php">
         <p> ERRO: Verifique se as datas de entrada e de saída
         estão corretas.</p>
         <?
         echo "<input type='hidden' name='estado'
         value='$estado'>";
```

```
         echo "<input type='hidden' name='cidade'
         value='$cidade'>";
         echo "<input type='hidden' name='tipoQuarto'
         value='$tipoQuarto'>";
         echo "<input type='hidden' name='qtd' value='$qtd'>";
         ?>
         <p align="center"><b><input type="submit" value="Voltar"
         name="B1"></b></p>
         </FORM>
      </BODY>
   </HTML>
<?}
```

18 Arquivo Resultado2.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0 Draft//EN">
<HTML>
   <HEAD><TITLE>Pesquisa de Hotéis</TITLE></HEAD>
   <BODY>
      <div align="center"><font size="4">
      <HR><b>Resultado da Pesquisa</b></font></div><HR>
      <FORM method="POST" action="Resultado3.php">
      <?
      printf("Nome: %s<br>\n", $nomes_hoteis[$contador]);
      // Consulta dos telefones do hotel corrente.
      $hotelcorrente = $cnpj_hoteis[$contador];
      $link = mysql_connect("localhost", "root") or die("Não foi
      possivel conectar o BD");
      mysql_select_db("ReservasOnLine",$link);
      $consulta_tel = "SELECT hot_tel_numero, hot_tel_tipo
      FROM hotel_telefone WHERE hot_tel_cnpj = '$hotelcorrente' ";
      $resultado = mysql_query($consulta_tel, $link);

      if (!$resultado)
      {
         echo("ERRO: " . mysql_error() . "\n$insercao\n");
         exit();
      }
      mysql_close($link);
      // Mostrar os telefones do hotel corrente.
      printf("Telefones: ");
      while($tupla = mysql_fetch_array($resultado))
      {
        echo $tupla["hot_tel_numero"], " - ",
        $tupla["hot_tel_tipo"], "<br>";
      }
      printf("Quarto: ");
      echo $qto_tipo[$contador], ", ", $qto_nopessoas[$contador],
```

```
      "Pessoa(s), R$", $qto_preco[$contador], "reais a diária, ",
      $qto_descricao[$contador];
      ?>
      <p>Reservar este quarto?</p>
      <input type="radio" value="V1" name="opcao"> Sim </p>
      <input type="radio" value="V2" checked name="opcao"> Não
      <?
      // Enviando os vetores com a pesquisa realiza anteriormente
      for ($i=0; $i<$num_hoteis; $i++)
      {
         echo "<input type='hidden' name='nomes_hoteis[]'
         value='$nomes_hoteis[$i]'>";
         echo "<input type='hidden' name='cnpj_hoteis[]'
         value='$cnpj_hoteis[$i]'>";
         echo "<input type='hidden' name='qto_numero[]'
         value='$qto_numero[$i]'>";
         echo "<input type='hidden' name='qto_tipo[]'
         value='$qto_tipo[$i]'>";
         echo "<input type='hidden' name='qto_nopessoas[]'
         value='$qto_nopessoas[$i]'>";
         echo "<input type='hidden' name='qto_preco[]'
         value='$qto_preco[$i]'>";
         echo "<input type='hidden' name='qto_descricao[]'
         value='$qto_descricao[$i]'>";
      }
      // Enviando as variaveis
      echo "<input type='hidden' name='contador'
      value='$contador'>";
      echo "<input type='hidden' name='num_hoteis'
      value='$num_hoteis'>";
      ?>
      <p align="center"><b><input type="submit" value="Avançar"
      name="B1"></b></p>
      </FORM>
   </BODY>
</HTML>
```

19 Arquivo Resultado3.php

```
<?
if ($opcao == V2)
{
   if ($contador+1 == $num_hoteis) // Se nao existe outro quarto
   para exibir.
   {?>
      <!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0
      Draft//EN">
      <HTML>
```

```
     <HEAD><TITLE>Reserva do Quarto</TITLE></HEAD>
     <BODY>
        <div align="center"><font size="4">
        <HR><b>Pesquisa de Hoteis</b></font></div><HR>
        <p>Obrigado pela consulta. Tenha um bom dia!</p>
     </BODY>
  </HTML>
<?}

else
{?>
   <!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0
   Draft//EN">
  <HTML>
     <HEAD><TITLE>Reserva do Quarto</TITLE></HEAD>
     <BODY>
        <div align="center"><font size="4">
        <HR><b>Resultado da Pesquisa</b></font></div><HR>
        <FORM method="POST" action="Resultado2.php">
        <p>Mais resultados...</p>
        <?
        // Enviando os vetores com mais resultados
        for ($i=0; $i<$num_hoteis; $i++)
        {
           echo "<input type='hidden' name='nomes_hoteis[]'
           value='$nomes_hoteis[$i]'>";
           echo "<input type='hidden' name='cnpj_hoteis[]'
           value='$cnpj_hoteis[$i]'>";
           echo "<input type='hidden' name='qto_numero[]'
           value='$qto_numero[$i]'>";
           echo "<input type='hidden' name='qto_tipo[]'
           value='$qto_tipo[$i]'>";
           echo "<input type='hidden' name='qto_nopessoas[]'
           value='$qto_nopessoas[$i]'>";
           echo "<input type='hidden' name='qto_preco[]'
           value='$qto_preco[$i]'>";
           echo "<input type='hidden' name='qto_descricao[]'
           value='$qto_descricao[$i]'>";
        }

        // Enviando as variaveis
        $contador++;
        echo "<input type='hidden' name='contador'
        value='$contador'>";
        echo "<input type='hidden' name='num_hoteis'
        value='$num_hoteis'>";
        ?>

        <p align="center"><b><input type="submit"
```

```
            value="Proximo"
            name="B1"></b></p>
            </FORM>
         </BODY>
      </HTML>
    <?}
}

else
{?>
   <!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0
   Draft//EN">
   <HTML>
      <HEAD><TITLE>Reserva do Quarto</TITLE></HEAD>
      <BODY>
         <div align="center"><font size="4">
         <HR><b>Reserva do Quarto</b></font></div><HR>
         <FORM method="POST" action="ConfirmaReserva.php">
         <p>Para confirmar a reserva do quarto, você deverá
         preencher os campos abaixo.</p>
         <br>CPF:
         <div>
         <td width="56%" height="21"><input type="text" name="cpf"
         size="12"></td>
         </div>
         <br>Senha:
         <div><td width="56%" height="21"><input type="password"
         name="senha" size="12"></td>
         </div><p>
         <?

         // Enviando os vetores com a pesquisa realiza
         anteriormente.
         for ($i=0; $i<$num_hoteis; $i++)
         {
            echo "<input type='hidden' name='nomes_hoteis[]'
            value='$nomes_hoteis[$i]'>";
            echo "<input type='hidden' name='cnpj_hoteis[]'
            value='$cnpj_hoteis[$i]'>";
            echo "<input type='hidden' name='qto_numero[]'
            value='$qto_numero[$i]'>";
            echo "<input type='hidden' name='qto_tipo[]'
            value='$qto_tipo[$i]'>";
            echo "<input type='hidden' name='qto_nopessoas[]'
            value='$qto_nopessoas[$i]'>";
            echo "<input type='hidden' name='qto_preco[]'
            value='$qto_preco[$i]'>";
            echo "<input type='hidden' name='qto_descricao[]'
            value='$qto_descricao[$i]'>";
```

```
        }

        // Enviando as variaveis
        echo "<input type='hidden' name='contador'
        value='$contador'>";
        echo "<input type='hidden' name='num_hoteis'
        value='$num_hoteis'>";
        ?>
        <p align="center"><b><input type="submit"
        value="Confirmar"
        name="B1"></b></p>
        </FORM>
      </BODY>
   </HTML>
<?}
```

20 Arquivo ConfirmaReserva.php

```
<?
$link = mysql_connect("localhost", "root") or die("Não foi
possivel conectar o BD");
mysql_select_db("ReservasOnLine",$link);
// Consulta da senha do cliente atraves do cpf digitado no
formulario anterior.
$consulta_senha = "SELECT cl_senha, cl_cpf FROM cliente WHERE
cl_cpf = '$cpf'";
$resultado = mysql_query($consulta_senha, $link);

if (!$resultado)
{
   echo("ERRO: " . mysql_error() . "\n$insercao\n");
   exit();
}
$tupla = mysql_fetch_array($resultado);
$senha_do_cliente = $tupla["cl_senha"];
$cpf_do_cliente = $tupla["cl_cpf"];

if ($senha == $senha_do_cliente && $senha != '')
{
   $insercao = "INSERT INTO reservas VALUES ($cpf,
   '$cnpj_hoteis[$contador]',
   $qto_numero[$contador], '$dtEntrada', '$dtSaida')";
   $resultado = mysql_query($insercao, $link);
   if (!$resultado)
   {
     echo("ERRO: " . mysql_error() . "\n$insercao\n");
     exit ();
   }
```

```
mysql_close($link);
$contador++;

if ($contador < $num_hoteis) // Continua a exibir os quartos
disponiveis.
{?>
   <!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0
   Draft//EN">
   <HTML>
      <HEAD><TITLE>Reserva do quarto</TITLE></HEAD>
      <BODY>
         <div align="center"><font size="4">
         <HR><b>Reserva do quarto</b></font></div><HR>
         <p>Reserva realizada com sucesso.</p>
         <FORM method="POST" action="Resultado2.php">

         <?
         // Enviando os vetores com a pesquisa realiza
         anteriormente
         for ($i=0; $i<$num_hoteis; $i++)
         {
            echo "<input type='hidden' name='nomes_hoteis[]'
            value='$nomes_hoteis[$i]'>";
            echo "<input type='hidden' name='cnpj_hoteis[]'
            value='$cnpj_hoteis[$i]'>";
            echo "<input type='hidden' name='qto_numero[]'
            value='$qto_numero[$i]'>";
            echo "<input type='hidden' name='qto_tipo[]'
            value='$qto_tipo[$i]'>";
            echo "<input type='hidden' name='qto_nopessoas[]'
            value='$qto_nopessoas[$i]'>";
            echo "<input type='hidden' name='qto_preco[]'
            value='$qto_preco[$i]'>";
            echo "<input type='hidden' name='qto_descricao[]'
            value='$qto_descricao[$i]'>";
         }

         // Enviando as variaveis
         echo "<input type='hidden' name='contador'
         value='$contador'>";
         echo "<input type='hidden' name='num_hoteis'
         value='$num_hoteis'>";
         ?>
         <p align="center"><b><input type="submit"
         value="Proximo" name="B1"></b></p>
         </FORM>
      </BODY>
   </HTML>
<?}
```

```
   else
   {?>
      <!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0
      Draft//EN">
      <HTML>
         <HEAD><TITLE>Reserva do quarto</TITLE></HEAD>
         <BODY>
            <div align="center"><font size="4">
            <HR><b>Reserva do quarto</b></font></div><HR>
            <p>Reserva realizada com sucesso. Tenha um bom
            dia!</p>
         </BODY>
      </HTML>
   <?}
}

else // Senha ou CPF invalidos.
{?>
   <!DOCTYPE HTML PUBLIC "-//W3C//DTD Compact HTML 1.0
   Draft//EN">
   <HTML>
      <HEAD><TITLE>Reserva do quarto</TITLE></HEAD>
      <BODY>
         <div align="center"><font size="4">
         <HR><b>Reserva do quarto</b></font></div><HR>
         <FORM method="POST" action="Resultado3.php">
         <?
         if ($cpf_do_cliente == '')
            echo("ERRO: CPF inválido ou não cadastrado. Por favor
            digite novamente");
         else
            echo("ERRO: Senha inválida. Por favor digite
            novamente");

         // Enviando os vetores com a pesquisa realiza
         anteriormente
         for ($i=0; $i<$num_hoteis; $i++)
         {
            echo "<input type='hidden' name='nomes_hoteis[]'
            value='$nomes_hoteis[$i]'>";
            echo "<input type='hidden' name='cnpj_hoteis[]'
            value='$cnpj_hoteis[$i]'>";
            echo "<input type='hidden' name='qto_numero[]'
            value='$qto_numero[$i]'>";
            echo "<input type='hidden' name='qto_tipo[]'
            value='$qto_tipo[$i]'>";
            echo "<input type='hidden' name='qto_nopessoas[]'
            value='$qto_nopessoas[$i]'>";
```

```
            echo "<input type='hidden' name='qto_preco[]'
            value='$qto_preco[$i]'>";
            echo "<input type='hidden' name='qto_descricao[]'
            value='$qto_descricao[$i]'>";
         }

         // Enviando as variaveis
         echo "<input type='hidden' name='opcao' value='V1'>";
         echo "<input type='hidden' name='contador'
         value='$contador'>";
         echo "<input type='hidden' name='num_hoteis'
         value='$num_hoteis'>";
         ?>
         <p align="center"><b><input type="submit" value="Voltar"
         name="B1"></b></p>
         </FORM>
      </BODY>
   </HTML>
<?}
```