PATRÍCIA APARECIDA PROENÇA

SisBDR – SISTEMA DE ACESSO A DIFERENTES BASES DE DADOS REMOTAS UTILIZANDO TECNOLOGIA MÓVEL

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do Curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

LAVRAS
MINAS GERAIS – BRASIL
2007

PATRÍCIA APARECIDA PROENÇA

SisBDR – SISTEMA DE ACESSO A DIFERENTES BASES DE DADOS REMOTAS UTILIZANDO TECNOLOGIA MÓVEL

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do Curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Área de concentração:

Engenharia de Software e Banco de Dados

Orientador:

Prof. Dr. Heitor Augustus Xavier Costa

LAVRAS
MINAS GERAIS – BRASIL
2007

Ficha Catalográfica preparada pela Divisão de Processo Técnico da Biblioteca Central da UFLA

Proença, Patrícia Aparecida

SisBDR – Sistema de Acesso a Diferentes Bases de Dados Remotas Utilizando Tecnologia Móvel/ Patrícia Aparecida Proença – Minas Gerais, 2007, 58p.

Monografia de Graduação – Universidade Federal de Lavras. Departamento de Ciência da Computação.

1. Aplicação Móvel. 2. Bases de Dados Remotas. I. PROENÇA, P. A. II. Universidade Federal de Lavras. III. Título.

PATRÍCIA APARECIDA PROENÇA

SisBDR – SISTEMA DE ACESSO A DIFERENTES BASES DE DADOS REMOTAS UTILIZANDO TECNOLOGIA MÓVEL

Monografia de graduação apresentada ao Departamento de Ciência da Computação da Universidade Federal de Lavras como parte das exigências do Curso de Ciência da Computação para obtenção do título de Bacharel em Ciência da Computação.

Aprovada em 17 / 03 / 2007

_____
Prof[a]. Msc. Olinda Nogueira Paes Cardoso

_____
Prof. Dr. Plínio de Sá Leitão Júnior

_____
Prof. Dr. Heitor Augustus Xavier Costa
(Orientador)

LAVRAS
MINAS GERAIS – BRASIL
2007

A Deus por sempre me iluminar e proteger, aos meus queridos pais José Márcio e Elza por todo amor, carinho e orações, ao meu irmão Márcio, meu grande amigo e companheiro. E a uma pessoa muito especial Sandro, por sempre estar ao meu lado.

AGRADECIMENTOS

Primeiramente agradeço a Deus por poder celebrar mais esta vitória.

A minha família pelo apoio, exemplo, motivação, segurança, amor e orações em todos momentos de minha vida.

Ao meu namorado, pelo carinho, companheirismo e apoio, que muito me ajuda a enfrentar momentos difíceis e também a viver grandes momentos felizes.

Agradeço a todos os meus familiares que mesmo distantes sempre me apoiaram.

Aos meus amigos, desde aqueles que mesmo de longe sempre mandavam aquela força até àqueles que batalharam, conviveram e festejaram comigo todo esse período de graduação. Os amigos os quais aqui nasceu a nossa amizade, nunca esquecerei de vocês, Flávia, Larissa, Jana, Cris, Inez, Keka, Nessa, Ale, Marcos e outros que aqui não citei.

A todos que me forneceram inestimável conhecimento além de apoio pessoal durante este percurso.

A todos professores, orientadores e membros da banca pela ajuda, ensinamentos e oportunidades oferecidos durante todo curso. Em especial, ao meu orientador, Prof. Heitor Augustus Xavier Costa, que me ajudou a construir a base de conhecimentos para realização deste trabalho. Aos funcionários do Departamento de Ciência da Computação pelo atenção e carinho.

A todos àqueles que de alguma forma contribuíram para a concretização desta Monografia.

SisBDR – SISTEMA DE ACESSO A DIFERENTES BASES DE DADOS REMOTAS UTILIZANDO TECNOLOGIA MÓVEL

RESUMO

Neste trabalho, é proposto um sistema de acesso a diferentes bases de dados remotas utilizando tecnologia móvel (SisBDR) que simula a perda de conexão durante a realização de consultas a servidor de banco de dados e, automaticamente, se conecta a um outro servidor de banco de dados, retornando o resultado das consultas sem erros e de maneira transparente ao usuário. A fim de que o SisBDR possua mobilidade, ele é implementado utilizando tecnologias como *WebServices* e J2ME. O SisBDR pode ser expandido de maneira que possa atender a outras operações de manutenção dos dados nos sistemas de banco de dados.

Palavras-Chave: Banco de Dados, Tecnologia Móvel, J2ME

ACCESS SYSTEM TO DIFFERENT REMOTE DATABASES USING MOBILE TECHNOLOGY

ABSTRACT

In this work, an access system to different remote databases is proposed using mobile technology (SisBDR), it simulates the loss of connection during the accomplishment of query in the database server and, automatically, connects to another database server, returning the result without errors and in transparent way to the user. The system was implemented using technologies like WebServices and J2ME aiming to improve its mobility. The system can be expanded so that he can assist to other operations of maintenance of data in database systems.

Keywords: Data base, Mobile Technology, J2ME

SUMÁRIO

LISTA DE FIGURAS

1. INTRODUÇÃO

A tecnologia da informação vem passando por uma grande revolução nos últimos anos, evoluindo consideravelmente dos primeiros computadores centrais até os atuais sistemas distribuídos. Com isso, os dispositivos móveis estão se tornando mais poderosos e, ao mesmo tempo, mais acessíveis aos consumidores, desde grandes multinacionais a pequenos usuários.

Um amplo espectro de pessoas utiliza a tecnologia móvel para fins de comunicação, trabalho, entretenimento, educação e outros motivos.

A mobilidade está ficando cada vez mais necessária na vida das pessoas devido aos inúmeros serviços que ela proporciona aos usuários. Os investimentos em tecnologias e em aplicações moveis têm sido um dos mais lucrativos nos últimos anos. O número de profissionais que utilizarão tecnologias móveis, como celulares, *handhelds* e *laptops*, deve crescer em mais de 20% nos próximos quatro anos, atingindo 878 milhões de pessoas, segundo um estudo da IDC (Instituto de Desenvolvimento Cultural) [IDC (2007)].

Os primeiros dispositivos criados que utilizavam redes sem fio, devido aos seus altos custos, eram ferramentas de finalidade especial para alguns trabalhadores móveis[1] e consumidores ricos[2]. A tecnologia móvel oferece conectividade que outros dispositivos não possuem e, por isso, tem sido cada vez mais utilizada para uso pessoal e profissional.

Segundo Rischpater (2001), a rede sem fio consiste em milhões de dispositivos pequenos e portáteis, prontamente disponíveis sempre e onde quer que um usuário necessite de informação. Computadores de mão, telefones inteligentes e dispositivos similares estão se tornando cada vez mais sofisticados com opções de conectividade sem fio. Muitos são os fins para utilizar essa tecnologia, tais como comunicação, trabalho, entretenimento, educação e localização.

A necessidade de disponibilizar informações cada vez mais rapidamente e onde quer que um usuário necessite faz com que o uso da rede sem fio cresça a cada dia. Em vários dispositivos, a rede sem fio se faz presente, desde simples telefones ou dispositivos similares a computadores portáteis.

---

[1] Trabalhadores móveis são pessoas que não têm lugar fixo de trabalho e necessitam comunicar-se com a empresa a qual presta serviços.

[2] Consumidores ricos são pessoas que possuem um poder aquisitivo alto.

Considerando o grande avanço da tecnologia móvel e o crescente número de finalidades para sua utilização, este trabalho tem o objetivo de desenvolver um sistema para dispositivos móveis. Este sistema simula a perda de conexão durante a realização de consultas a um servidor de banco de dados e, automaticamente, se conecta a um outro servidor de banco de dados, retornando o resultado das consultas sem erros e de maneira transparente ao usuário.

1.1. Motivação

A expectativa de crescimento do desenvolvimento de aplicações para dispositivos móveis é grande, pois estes dispositivos estão se tornando cada vez mais populares. A cada dia, mais pessoas utilizam a mobilidade, pelo fato da conveniência e das numerosas funções oferecidas.

Conforme Lee *et al.* (2005), as diversas funções presentes nos dispositivos móveis foram implementadas na forma de aplicações móveis. Por exemplo, relógio, jogos, calculadora, calendário, correio eletrônico, agenda, contatos, tarefas, notícias, serviço de bancos e MSN[3].

Muitas empresas têm investido para aumentar a capacidade de conectar e tornar a mobilidade acessível às pessoas. A necessidade de disponibilizar informações cada vez mais rapidamente faz com que as empresas passem a investir na mobilidade. A interação entre os profissionais, através do uso de dispositivos móveis, colabora nas tomadas de decisão, fazendo com que essas ocorram em um curto espaço de tempo, proporcionando uma grande contribuição no aumento da lucratividade das empresas.

O telefone celular é um exemplo perfeito de um dispositivo sem fio. A crescente demanda pública de telefones celulares durante os últimos anos resultou em uma das indústrias mais bem sucedidas deste século. O celular se faz necessário na vida de uma grande parte da população mundial, para fins profissionais e pessoais.

Inserido na conjuntura atual, em que a utilização de dispositivos móveis tornou-se quase indispensável e, acima de tudo, acessível à maioria das pessoas, o trabalho consiste no desenvolvimento de uma aplicação móvel.

[3] MSN (*Messenger*) é um software para interatividade em tempo real.

1.2. Objetivos

A mobilidade, atualmente, é uma das tecnologias de maior investimento nos últimos anos e, com isso, surge a necessidade de aplicações suportarem essa tecnologia. Pensando nisso, o objetivo do trabalho é um estudo, através de pesquisas, sobre as tecnologias utilizadas no desenvolvimento de aplicações para dispositivos móveis e o desenvolvimento de uma aplicação para um dispositivo móvel.

A aplicação desenvolvida no trabalho visa a criação de um sistema para um dispositivo móvel, que simule a perda de conexão durante consultas a servidor de banco de dados e, automaticamente, se conecte em outro servidor de banco de dados para que as consultas solicitadas sejam realizadas com êxito.

A idéia de tratar possíveis perdas de conexão com um servidor de banco de dados vem da grande importância da mobilidade no dia-a-dia de muitos usuários. Com a utilização de um dispositivo móvel, o usuário pode se mover e, ao mesmo tempo, realizar consultas, sem que a perda de conexão com o servidor de banco de dados seja um problema visível, ou seja, a perda de conexão é transparente ao usuário. O sistema permite a conexão a dois servidores de banco de dados que possuem as mesmas informações armazenadas (redundância de informações), proporcionando eficientes consultas mesmo havendo perda de conexão a um dos servidores.

1.3. Metodologia

1.3.1. Tipo de Pesquisa

Metodologia é o método usado para conduzir uma pesquisa. Neste sentido, a metodologia pode ser vista como conhecimento geral e habilidade que são necessários ao pesquisador para se orientar no processo de investigação, tomar decisões oportunas, selecionar conceitos, hipóteses, técnicas e dados adequados [Thiollent (1994)].

As pesquisas exploratória e experimental foram os métodos de pesquisa utilizados no desenvolvimento deste trabalho.

Segundo Gil (1991), a pesquisa exploratória visa a descoberta, o achado, a elucidação de fenômenos ou a explicação de fenômenos que não eram aceitos apesar de evidentes.

A pesquisa exploratória tem o objetivo de descobrir teorias e práticas que modificarão as existentes, obter alternativas ao conhecimento científico convalidado e, principalmente, inovar tecnologias. A pesquisa exploratória trata do levantamento da bibliografia publicada e que tenha relação à pesquisa a ser realizada. No caso deste trabalho, o levantamento bibliográfico foi relacionado a aplicações móveis, tecnologia *J2ME* e *WebServices*. Assim sendo, a pesquisa exploratória tem a finalidade de colocar os pesquisadores/leitores em contato direto com referências bibliográficas relevantes do assunto.

A pesquisa experimental busca ensaios e estudos de laboratório, modelagem, simulação, sistemas e circuitos. O objetivo da pesquisa experimental é a descoberta de novas matérias, métodos, técnicas, protótipos de *software*. Na pesquisa experimental, os pesquisadores desempenham um papel ativo no equacionamento do problema considerado como central na pesquisa, no acompanhamento e na avaliação das ações desencadeadas em função dos problemas, atuando para encontrar soluções [Gil (1991)]. O motivo pela escolha da pesquisa experimental é devido a sua grande capacidade de resolver problemas específicos, além de ser um método adaptável que auxilia pesquisadores a lidar com inserção de conhecimentos na prática, no presente caso o desenvolvimento de um sistema de acesso a diferentes servidores de banco de dados com reconexão automática utilizando tecnologia móvel.

1.3.2. Procedimentos Metodológicos

O trabalho realizado foi o estudo e o desenvolvimento de um sistema de acesso a diferentes servidores de banco de dados com reconexão automática utilizando tecnologia móvel. Este trabalho inicialmente pesquisou sobre as tecnologias utilizadas no desenvolvimento de aplicações móveis e, em seguida, foi desenvolvido o sistema.

A parte de pesquisa exploratória foi desenvolvida em livros textos relacionados ao desenvolvimento de aplicações móveis e no Laboratório do Departamento de Ciência da Computação da Universidade – DCC/UFLA utilizando a *Internet*.

No desenvolvimento, um computador foi utilizado com a seguinte configuração: *AMD Athlon (TM) XP 2600* de *1.92 GHz*, com *512 Mb* de memória *RAM* e *HD* de *80 GB*. O sistema operacional foi o *Microsoft Windows XP Versão 2002*.

Para a modelagem do sistema, foi utilizada a ferramenta CASE (*Computer Aided Software Engineering*) *JUDE Community 3.1.1* (*Model Version: 23*), onde foram desenvolvidos os diagramas envolvidos no desenvolvimento do trabalho.

Na implementação, foi utilizada a tecnologia J2ME (*Java 2 Micro Edition*). O uso desta tecnologia se justifica pela portabilidade entre plataformas de *hardware* e *software*, gratuidade e tratamento de exceções e de componentes de interfaces gráficas [Almeida (2004)]. Estas características tornam a tecnologia J2ME apropriada para implementação do sistema de acesso a diferentes servidores de banco de dados com reconexão automática. Foi utilizado o IDE (*Integrated Development Environment*) *NetBeans 5.5 Beta 2* e o pacote *Netbeans Mobility-5_5*, pois permite a criação de aplicações móveis pelo *NetBeans*.

Por se tratar de uma aplicação para um dispositivo móvel e devido a algumas limitações (capacidade de processamento, tamanho da tela e baixa fonte de energia) desses dispositivos para a implementação das operações com o servidor de banco de dados, foi necessária a utilização da tecnologia JSP (*Java Server Pages*). Esta tecnologia possui as mesmas características apresentadas pela tecnologia J2ME e algumas outras a considerar, tais como possibilidade de usar recursos de processamento de banco de dados e tratamento de exceções e de componentes de redes cliente/servidor baseadas na *Internet* [Freire (2002)]. Para servidor de JSP, foi utilizado o *Tomcat*, por ser gratuito, fácil de configurar e atender às necessidades durante a fase de desenvolvimento do sistema.

A parte servidor e a parte cliente (aplicação móvel) foram tratadas pelo *NetBeans*. Para armazenamento dos dados, foi utilizado o SGBD (Sistema Gerenciador de Banco de Dados) *MySQLServer* 5.0. O *MySQLServer* foi escolhido para ser o Sistema de Gerenciamento de Banco de Dados (SGBD), por ser confiável, robusto, gratuito, estável, portável para diferentes plataformas, oferecer um alto nível de segurança e ser de fácil utilização em conjunto com o J2ME.

1.4. Estrutura do Trabalho

O trabalho está estruturado da seguinte forma:

O Capítulo 2 apresenta algumas considerações sobre Aplicações Móveis, bem como as vantagens e as desvantagens, a importância, a arquitetura e o desenvolvimento de aplicações móveis.

O Capítulo 3 mostra sucintamente a especificação J2ME, a interface com o usuário e a estrutura de conexão.

O Capítulo 4 apresenta uma breve descrição de *WebServices*, bem como o seu funcionamento, a sua arquitetura e as tecnologias que compõe os *WebServices*.

O Capítulo 5 descreve o sistema SisBDR, o seu funcionamento, a sua arquitetura, a sua modelagem e uma apresentação de um estudo de caso utilizado no SisBDR.

O Capítulo 6 coloca algumas considerações finais, contribuições do trabalho e sugestões de trabalhos futuros.

2. APLICAÇÕES MÓVEIS

2.1. Considerações Iniciais

Com a miniaturização dos dispositivos móveis, novas maneiras para os usuários interagirem com os seus computadores foram fornecidas. Com isso, houve grande necessidade de desenvolvimento de aplicações móveis para melhor conforto desses usuários.

As aplicações desenvolvidas para dispositivos móveis ganharam espaço no mundo sem fio, como bancos, mercado financeiro, notícias e área médica.

Segundo Lee *et al.* (2005), uma aplicação móvel antes de ser projetada, desenvolvida e implantada deve ser planejada de forma correta e eficiente. Ela é implementada por razões de negócio (por exemplo, melhorar a produtividade e aumentar a precisão) além de outras razões, tais como comunicação e entretenimento.

O desenvolvimento das aplicações móveis, geralmente, é baseado em aplicações que existem e que precisam se tornar móveis devido ao grande uso da mobilidade.

No desenvolvimento de aplicações móveis, devem ser considerados alguns fatores, tais como: mobilidade, contexto de negócio, arquiteturas, infra-estrutura móvel, interface com o usuário de cliente móvel, aplicações clientes móveis, transferência de dados cliente-servidor, segurança e gerenciamento do desenvolvimento.

2.2. Vantagens e Desvantagens

Diversas e excelentes são as razões para usar aplicações móveis hoje em dia. Algumas afetam diretamente a vida particular de seus funcionários, enquanto outras, a sua interação com os clientes e com as aplicações de negócios existentes, da mesma forma que outras afetam a parte financeira do negócio [Rischpater (2001)].

Existem vantagens na utilização de novas aplicações móveis. Sob a ótica da operadora, novos serviços equivalem a mais receitas dos mesmos assinantes. A luta por extrair mais dinheiro destes assinantes está apenas começando no Brasil, mas é fortemente travada especialmente na Europa e Ásia, onde os elevados níveis de penetração de telefonia tornam muito difícil a aquisição de novos clientes. Diferenciação via serviços tornou-se um dos poucos modos de tirar os clientes de outras operadoras e manter os

próprios clientes felizes. Sob a ótica do usuário final, seja ele pessoa física ou corporativo, existem diversas vantagens advindas das tecnologias móveis, acesso a informações de necessidade instantânea em tempo real em qualquer lugar e de forma simples e rápida [Castaldelli (2003)].

Segundo Lee *et al.* (2005), algumas vantagens podem-se destacar:

- As aplicações móveis contribuem na vida pessoal e profissional das pessoas. Por exemplo, através de um celular, os pais podem saber notícia de seus filhos e as pessoas podem se comunicar com algum socorro para o carro que quebrou no meio de uma viagem de negócios ou de passeio;
- Aumentar a flexibilidade e a acessibilidade dos funcionários, fornecendo soluções móveis. Um vendedor, por exemplo, pode se comunicar com a empresa no ato da venda para consultar produtos, preços e dados do cliente, melhorando o tempo e a eficiência da venda;
- Fornecendo informações atualizadas ao funcionário, irá aprimorar a sua segurança. Como exemplo, tem a execução de um serviço em uma mina, o funcionário que se encontra dentro dela pode não perceber que ela está desabando e, através de um dispositivo móvel, o funcionário é avisado e poderá deslocar-se mais rápido;
- A melhoria da eficiência do fluxo de trabalho e da produtividade pode ser obtida através do uso de aplicações móveis. Por exemplo, um vendedor normalmente anota os pedidos de compra em um papel e, depois, passa para o computador para ser feita a solicitação na empresa. Isso pode estar sujeito a erros, pois precisa ser feito duas vezes. Com a utilização de uma aplicação móvel, o funcionário precisará fazer só uma vez, reduz a quantidade de tarefas repetitivas permitindo ao funcionário ser mais eficiente e produtivo;
- Melhorar a atualização e a precisão de dados é um dos benefícios do uso de aplicações móveis, pois uma força de trabalho mobilizada pode receber e fornecer informações para os sistemas de negócios existentes de forma oportuna. O número de erros também pode reduzir durante a coleta de dados e no processo de criação dos relatórios. Portanto, a atualização e a precisão dos dados são melhoradas;
- Com as aplicações móveis, é possível aprimorar os processos de negócios. Por exemplo, uma equipe de trabalho que dispõe da tecnologia móvel como um canal adicional que fornece e que recebe dados de/para os sistemas de negócios pré-

existentes. Agindo assim, as empresas tornam-se capazes de descobrir melhorias e eliminar redundâncias nos fluxos de trabalho existentes;

- As empresas podem utilizar dispositivos móveis para auxiliá-las a localizar, a rastrear e a monitorar equipamentos e outros itens. Com isso, poderá melhorar o controle de inventário e os custos de lançamento;
- A satisfação do cliente é obtida quando os processos de serviços se tornam mais eficientes e rápidos. Isso, por sua vez, pode levar a um incremento na rentabilidade da empresa.

Existem situações nas quais podem ser observados alguns pontos fracos. Rischpater (2001) apresenta algumas razões de negócios, sociais e de privacidade social e ambiental que devem ser levadas em consideração, a saber:

- Considerações de negócio. A implantação da mobilidade pode ter um alto custo. Antes de investir na tecnologia, a empresa deverá ter um entendimento completo do lucro ou prejuízo que poderá ser retornado a empresa. Os fatores que contribuem para um alto custo do investimento são: custo de *hardware* e *software*, custo de comunicação, custo de desenvolvimento e implantação, ruptura de serviços existentes, custo de operações e custos diversos;
- Considerações sociais. A utilização de dispositivos móveis passou de simplesmente um recurso em caso de emergência para um meio de dependência ou um meio de obsessão;
- Privacidade e considerações de segurança. Embora a utilização de dispositivos móveis seja algo muito conveniente, é necessário que se leve em consideração as questões de privacidade e segurança na transferência de dados. Os dispositivos móveis não são totalmente seguros, pois podem ser perdidos e os dados armazenados neles podem ser comprometidos se as precauções necessárias para mantê-los em segurança não forem tomadas;
- Considerações ambientais. Os dispositivos móveis apesar de serem pequenos não são descartáveis, seus componentes são tóxicos. Muitas substâncias que se encontram dentro dos dispositivos estão relacionadas a diversos tipos de câncer e a distúrbios neurológicos e podem impor sérios problemas ao meio ambiente.

2.3. Importância

Mesmo com a crise econômica e a crise política que o país está enfrentando, são feitos muitos investimentos em infra-estrutura para o aumento da produção e redução de

custos nos processos de trabalho. Estes investimentos são necessários para uma possível inserção de empresas nacionais no mercado internacional, com oportunidades de exportação e difusão de produtos e serviços nacionais no exterior.

Segundo Compera (2006), os investimentos em Tecnologia da Informação são potencial gerador de receita para o mercado atualmente, tanto para os produtores de tecnologia, quanto para as empresas que utilizam recursos inovadores nos processos de produção, passando inclusive pelo mercado investidor, que é responsável pela difusão de novas tecnologias no mercado.

Para complementar este potencial de crescimento, grandes empresas nacionais estão em produção de soluções que auxiliam no controle da comunicação e da transmissão de informações, como estoque, vendas, distribuição, informações financeiras para bancos e informações operacionais no setor de saúde.

Que os sistemas de computação e os sistemas de comunicação estão se tornando incrementalmente interdependentes é evidente em muitas áreas da sociedade. Essa tendência força a comunidade de computação não só a desenvolver sistemas inovadores, mas a redefinir os sistemas existentes em termos dos papéis que estes representam.

Dentro deste contexto, a computação móvel está aumentando em importância e presença. O desenvolvimento é o resultado de avanços tecnológicos em muitos domínios. Prevê-se que redes de comunicação sem fio, combinadas com o desenvolvimento de computadores portáteis tais como, *notebook* e PDA (*Personal Digital Assistant*), permitirão ao usuário deslocar-se junto com seu ambiente computacional e ter um acesso constante às fontes de informações [DEVMEDIA (2006)].

2.4. Arquitetura

As arquiteturas de aplicação móveis geralmente são modeladas para destacar ou ilustrar o *layout* do *software* (código de aplicação e plataforma) e *hardware* (cliente, servidor e dispositivos de rede). Existem muitas combinações possíveis de *software* e *hardware*, mas as arquiteturas de aplicação freqüentemente são encaixadas em uma série de padrões reconhecíveis.

Conforme Lee *et al.* (2005), as arquiteturas de aplicação são modeladas em termos de uma arquitetura cliente-servidor, onde um ou mais dispositivos clientes solicitam informações a um servidor e a distribuição do código da aplicação é feita em camadas.

O código de uma aplicação não é necessariamente uniforme na aplicação. Algumas seções do código são mais adequadas para tratar a interface com o usuário, enquanto outras seções são desenvolvidas com a finalidade de gerenciar a lógica do negócio ou comunicar-se com o banco de dados.

Segundo Rischpater (2001), as camadas são módulos de códigos colocados em pastas ou diretórios diferentes no cliente ou no servidor. A divisão em camadas indica a divisão de funções dentro do código da aplicação em uma única máquina.

Um bom projeto de *software* ajuda na capacidade de reutilização do código. No desenvolvimento de um bom projeto, o código da aplicação é dividido em camadas, sendo uma questão de segurança e conveniência. No código do lado do cliente, geralmente, existem de zero a três camadas de código da aplicação. Do lado do servidor, no código há normalmente de uma a três camadas de código de aplicação.

Um cliente magro (*Thin Client*) não tem camada de código, basicamente ele não tem código da aplicação personalizado. Esse tipo de cliente é possível utilizar na arquitetura cliente-servidor, se o servidor armazenar o código personalizado da aplicação. Um cliente com uma a três camadas de código da aplicação é comumente denominado cliente gordo (*Fat Client*) (Rischpater, 2001).

Um servidor pode ter de uma a três camadas de código da aplicação personalizadas e, por definição, não se pode ter a camada de código em um servidor.

Segundo Lee *et al.* (2005), a camada de apresentação (primeira camada) interage com o usuário e é a camada que trata a parte de interface de uma aplicação. A outra camada (segunda camada) é denominada camada de negócio, pois trata da lógica comercial e da interação entre os dados. Uma outra camada (terceira camada) é denominada camada de acesso a dados e trata da comunicação com o meio persistente. O número de camadas pode ser mais que três em qualquer cliente ou servidor, porém muitas camadas podem se tornar difíceis de manejar e de gerenciar. Por isso, esse tipo de arquitetura é raramente utilizado.

Os dispositivos móveis em geral operam em um de três modos [Lee *et al.* (2005)]:

- Sempre conectado. Por exemplo, uma empresa pode possuir uma rede sem fio e um conjunto de aplicações e de servidores que permitam aos empregados permanecerem

conectados e utilizarem seus dispositivos móveis (*PDAs*[4], *Tablet PCs*[5] e *PCs laptop*[6]) enquanto estiverem nas dependências da empresa;

- Parcialmente conectado. Por exemplo, um funcionário móvel da empresa pode se conectar periodicamente a um servidor no escritório para receber mensagens de correio eletrônico, obter informações ou a lista de tarefas a serem realizadas e desconectar o dispositivo móvel. O usuário pode atualizar as informações localmente em dispositivos móvel antes de reconectar mais tarde e de ressincronizar o dispositivo móvel com o servidor;
- Nunca conectado. Diversos dispositivos nunca se conectam a sistemas *back-end*[7]. Como por exemplo, dispositivos de jogo.

O tipo de conexão que o dispositivo utiliza necessita de uma forma especial de sincronização dos dados entre o dispositivo móvel e os sistemas *back-end*. A sincronização pode ser realizada de duas formas:

- Continuamente. Na forma contínua de conectividade entre o cliente e o servidor, as sincronizações de dados são contínuas e podem ser obtidas por meio síncrono (os dados são completamente armazenados antes que o servidor confirme o recebimento dele e libere a interface com o cliente) ou assíncrono (os dados não precisam ser armazenados completamente antes que o servidor dê a confirmação ao cliente, o servidor confirma imediatamente a solicitação e somente depois executa a solicitação de armazenamento, iniciando uma conversa para informar ao cliente que está pronto) [Lee *et al.* (2005)];
- Através de um método de armazenamento e encaminhamento. A conectividade entre cliente e servidor pode não ser garantida, neste caso é possível armazenar e transmitir as informações de forma segura utilizando um método chamado armazenar e encaminhar. Supondo, por exemplo, que um usuário queira inserir algum dado enquanto seu dispositivo não está conectado no servidor. Uma aplicação cliente móvel

[4] O termo PDA (*Personal Digital Assistant*) foi elaborado pela *Apple Computer*. Originalmente, pretendia-se que o PDA fosse uma versão eletrônica de uma biblioteca pessoal, possuindo relógio, calendário para compromissos de negócios, lista de tarefas e relação de telefones para a informação de contato. Mas com a evolução dos PDAs outras funcionalidades foram incluídas tais como correio eletrônico, acesso à *Internet*, jogos, dentre outras [Lee *et al.* (2005)].

[5] Um *Tablet PC* é um computador móvel de um uso geral integrado a uma grande tela interativa. Talvez seu recurso mais atraente seja a sua capacidade de permitir aos usuários escrever direta e confortavelmente sobre a tela usando uma caneta [Lee *et al.* (2005)].

[6] Um PC *laptop* é um computador móvel de uso geral com uma grande tela integrada que apresenta um mouse e um teclado como dispositivos típicos de entrada de dados [Lee *et al.* (2005)].

[7] *Back-End* é uma função default onde ficam, por exemplo, *mailboxes* dos usuários. Todo servidor não configurado para ser um *front-end* é automaticamente um *back-end server*.

pode armazenar os dados em um banco de dados local e mais tarde, quando uma conexão for estabelecida, a aplicação encaminhará os dados do banco de dados local para o do servidor. Armazenar e encaminhar é um método poderoso que permite aos usuários móveis a capacidade de trabalhar, mesmo quando eles não estão conectados a um servidor [Lee *et al.* (2005)].

Os padrões parcialmente conectados são atualmente os mais dominantes, pois a conectividade nem sempre pode ser garantida. Espera-se que no futuro, os padrões sempre conectados venham a ser os mais utilizados.

Alguns princípios devem ser respeitados na construção de uma boa arquitetura de projeto. Na prática, nem sempre é possível implementar estes princípios, contudo muitas das melhores arquiteturas de aplicação móvel obedecem a vários desses princípios [Rischpater (2001)], são eles:

- Requisitos (o projeto deve abranger os requisitos funcionais, de negócio e do usuário);
- Independência de tecnologia (desenvolver aplicações móveis que sejam independentes de dispositivos e de plataformas);
- Alto desempenho e disponibilidade (a arquitetura deve ter desempenho excelente durante períodos normais e de pico de demanda de recursos);
- Escalabilidade (a arquitetura deve ser escalável para acomodar aumento no número de usuários, aplicações e funcionalidade);
- Requisitos de sistema de usuário (o sistema deve tratar o mais amplo espectro e número de usuários possível).

2.5. Desenvolvimento

O desenvolvimento de aplicações móveis, executadas em diferentes ambientes, traz novos desafios para prover a funcionalidade que os usuários móveis esperam obter do sistema.

Segundo Compera (2006), o dinamismo fornecido pelo ambiente permite ao usuário deslocar-se enquanto a aplicação continua em execução. Durante o deslocamento, as aplicações estão sujeitas a importantes variações no ambiente de execução (banda, latência e serviços), as quais devem ser absorvidas por um comportamento adaptativo. A complexidade inerente deste ambiente (heterogêneo, dinâmico e adaptativo) induz à necessidade de prover suporte para o desenvolvimento de aplicações.

As aplicações móveis possuem três estados (iniciado, pausado e destruído) durante o ciclo de vida dessas aplicações e representam, respectivamente, os momentos em que as aplicações começam a rodar, entram no estado de pausa ou são finalizadas. As aplicações entram no estado de pausa quando, durante a execução das aplicações, o dispositivo móvel recebe um chamado de algum evento com maior prioridade do que a aplicação em execução, por exemplo, o recebimento de uma chamada telefônica. Além disso, uma aplicação pode entrar no estado de pausa a pedido do usuário. O estado destruído é acionado quando a aplicação é finalizada, fato que pode ocorrer a pedido do usuário que está executando a aplicação ou quando ocorre algum erro desconhecido ou não tratado pelo programador e que implica na finalização abrupta da aplicação.

No desenvolvimento de uma aplicação móvel, é criada uma classe abstrata que define os métodos necessários para gerenciar o ciclo de vida das aplicações móveis e a aplicação deve ter uma (e somente uma) classe que implementa os três principais estados da aplicação [Devmedia (2006)].

Uma aplicação móvel é projetada, desenvolvida e implantada dentro de um contexto. São implementadas por razões de negócio, como aumento da produção e da precisão. Geralmente, são integradas a aplicações desenvolvidas.

Conforme Lee *et al.* (2005), alguns fatores devem ser analisados no desenvolvimento de uma aplicação móvel, tais como: mobilidade, contexto de negócio, arquiteturas de aplicação móvel, infra-estrutura móvel, interface com o usuário, aplicações clientes móveis, transferência de dados cliente-servidor, as arquiteturas de aplicações existentes, segurança, gerenciamento do desenvolvimento e estudos de caso.

2.6. Considerações Finais

A utilização de dispositivos móveis tornou-se quase indispensável e, acima de tudo, acessível à maioria das pessoas. Atualmente, milhares de pessoas utilizam a tecnologia móvel, desde usuários com fins comerciais a crianças para jogos, mensagens, *Internet*, comunicação, dentre outros.

Apesar, de ser uma tecnologia recente, muitos projetos vêm sendo desenvolvidos com relação a aplicações móveis. A cada dia mais empresas estão surgindo no mercado da mobilidade. Percebe-se que no momento vem sendo um dos negócios mais bem sucedidos da economia mundial.

A tecnologia móvel, embora sendo uma área de muito investimento e estudo, ainda não é muito eficiente em termos de portabilidade, funcionalidade, usabilidade ou conectividade. Alguns dispositivos apesar de serem leves, ainda são grandes e incômodos e, além disso, não são capazes de funcionarem por muito tempo, precisando ser recarregados.

3. PLATAFORMA J2ME

3.1. Considerações Iniciais

Nos últimos anos, os dispositivos móveis estão ficando mais sofisticados, apresentando funcionalidade que muitas vezes os próprios usuários não sabem como explorá-las. Com o objetivo de atender às necessidades de clientes cada vez mais exigentes, as indústrias de tecnologia se superam a cada dia com *software* melhor elaborado e *hardware* mais sofisticado.

Devido a grande explosão no mercado de comunicação móvel juntamente com a capacidade de executar programas escritos na linguagem de programação Java em telefones celulares e outros dispositivos móveis, surge a necessidade de profissionais capacitados em desenvolver sistemas para tais dispositivos [Caldeira (2005)].

Com base na grande evolução da comunicação móvel, o uso de J2ME (*Java 2 Micro Edition*) para desenvolvimento de aplicações para dispositivos móveis vem sendo cada vez mais incentivado por fabricantes de celulares e operadoras de telefonia e a quantidade de dispositivos que suportam essa tecnologia cresce a cada dia. Segundo Almeida (2004), um dos motivos para utilizar esta tecnologia é os desenvolvedores serem livres para desenvolver aplicações e executá-las em qualquer dispositivo de qualquer fabricante que possua uma máquina virtual.

Hoje é possível, mesmo em equipamentos de baixa capacidade de processamento, como celulares de preço médio, disponibilizar aplicações comerciais ou gerenciais eficazes. Sendo assim, os usuários ficam livres para escolher o modelo de dispositivos móveis mais adequados às suas necessidades que possam executar os aplicativos desejados.

3.2. Entendendo J2ME

A linguagem de programação Java vem sofrendo transformações desde o seu lançamento. O aumento no número de aplicações e, conseqüentemente, no número de bibliotecas padrão da linguagem, levou à criação de três divisões na plataforma a partir da versão dois da linguagem: J2SE (*Java 2 Standard Edition*), J2EE (*Java 2 Enterprise Edition*) e J2ME. Essas divisões são chamadas por alguns de ambientes de desenvolvimento, distribuições principais ou edições [Muchow (2004)].

A edição mais popular é conhecida como J2SE seu uso é voltado para PCs e servidores, onde há grande necessidade de aplicações.

A segunda distribuição da linguagem, denominada J2EE, foi desenvolvida para atender aplicações que demandam grande robustez e segurança, sendo muitas vezes executadas em servidores de grande capacidade, por exemplo, aqueles voltados para serviços de comércio eletrônico [Topley (2002)].

Para o mercado constituído por dispositivos de menor capacidade computacional, foi lançada a plataforma J2ME.

Embora existisse a API (*Application Program Interface*), J2SE em um micro dispositivo não era realístico. Por exemplo, a tela de um celular é limitada, não pode fornecer a funcionalidade disponível no *Abstract Window Toolkit* (a primeira interface gráfica com o usuário lançada com Java). Para suprir essas necessidades especiais dos dispositivos para o consumidor que estão fora da abrangência do J2SE e do J2EE, foi lançada a *Java Micro Edition* [Luz *et al.* (2005)].

A Figura 3.1 ilustra algumas plataformas que implementam a linguagem Java.

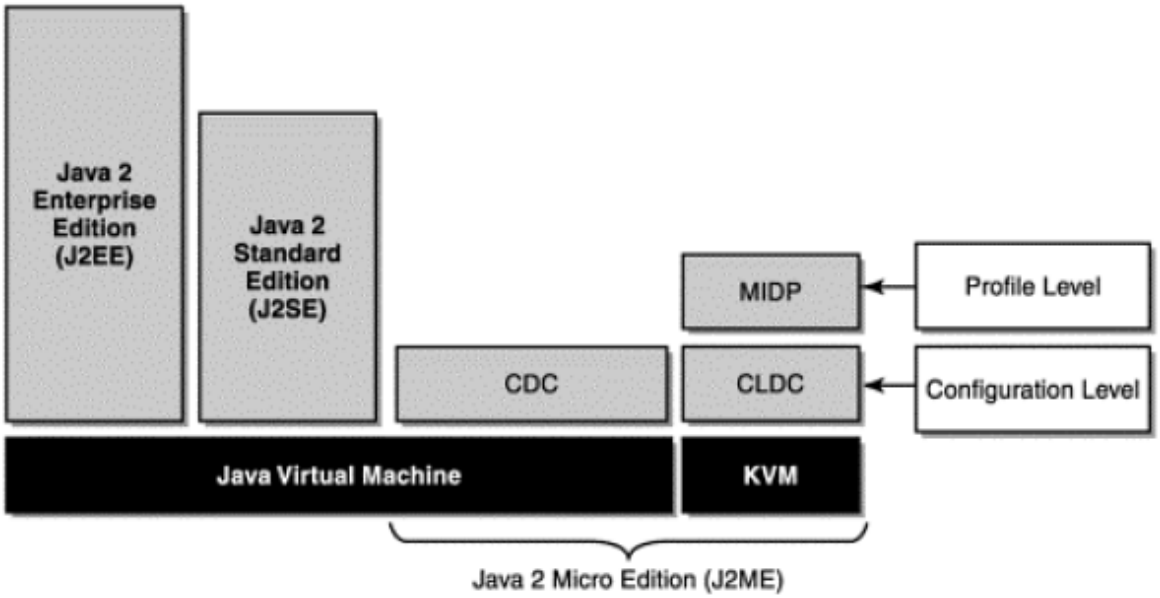

Figura 3.1 – Edições da linguagem Java [Fonte: Muchow (2004)]

Conforme Topley (2002), J2ME é uma versão reduzida do J2SE, foram feitas remoções e algumas modificações em partes fundamentais do J2SE para que houvesse a possibilidade de rodar a linguagem em um dispositivo com capacidade limitada. A edição é uma API Java que veio suprir as necessidades de um mercado cada vez maior de dispositivos computacionais que vão desde *pagers* e celulares a aparelhos domésticos

como televisores digitais, refrigeradores e automóveis. Assim sendo, um dos principais objetivos da plataforma J2ME é definir soluções que sejam válidas para essas tecnologias e padrões.

Muitos dos dispositivos móveis não possuem opção de *download* e *software* de instalação, os que possuem foram configurados durante o processo de fabricação. Com a utilização de uma implementação de J2ME em um dispositivo, é possível navegar, fazer *download* e instalar aplicativos Java [Gomes (2007)]. A Figura 3.2 ilustra alguns dispositivos que suportam a tecnologia J2ME.

Figura 3.2 – Dispositivos J2ME [Fonte: Gomes (2007)]

Cada uma das edições Java define um conjunto de tecnologias que podem ser utilizadas para o desenvolvimento de aplicações. São partes da plataforma Java a especificação de uma Máquina Virtual, um conjunto de classes relacionadas e as ferramentas necessárias à instalação e configuração de aplicações.

3.3. Arquitetura J2ME

Para que a edição J2ME possa acomodar uma ampla variedade de dispositivos eletrônicos de consumidor e dispositivos incorporados, ela é dividida em *Configurations* (configurações), *Profiles* (perfis) e APIs opcionais. Devido à essa divisão, o desenvolvedor pode conhecer informações específicas sobre as diferentes famílias de dispositivos e as APIs disponíveis em cada uma delas.

Com o objetivo de resolver os problemas de sub-aproveitamento de recursos, a tecnologia J2ME foi divida em alguns grupos, cada um voltado a um porte de dispositivos. A esta divisão dá-se o nome de *Configuration* [Gomes (2007)].

Configuration define o mínimo que um desenvolvedor pode esperar do dispositivo, classificando-os por capacidade de memória e processamento. Foi introduzida pela Sun para suportar uma ampla variedade de produtos que se encaixam no escopo de J2ME. A plataforma J2ME possui atualmente duas configurações principais a CDC (*Connected Device Configuration*) e a CLDC (*Connected Limited Device Configuration*) [Almeida (2004)].

A Figura 3.3 apresenta a divisão de alguns dispositivos segundo a sua capacidade computacional.

Figura 3.3 – Classificação dos dispositivos segundo sua capacidade computacional [Fonte: Gomes (2007)]

A fim de prover a plataforma J2ME com um ambiente no qual aplicações pudessem ser executadas, foi desenvolvida uma máquina virtual específica denominada KVM (*Kilobyte Virtual Machine*). A máquina virtual foi desenvolvida para executar em dispositivos dotados de um processador de 16 ou 32 *bits* e que não dispõem de mais que algumas centenas de *kilobytes* de memória.

A primeira *Configuration*, denominada CLDC, é um conjunto de classes mais apropriado para dispositivos pequenos, dispositivos dotados de mecanismos de transmissão de dados e com limitações de desempenho e memória (entre 160KB e 500KB de memória disponível). A CLDC consiste em uma máquina virtual reduzida KVM. A tela é de tamanho pequeno e a fonte de energia é reduzida e fornecida por baterias. Exemplos típicos incluem telefones celulares digitais, *pagers*, dispositivos de áudio e vídeo portáteis e pequenos terminais de consulta ou pagamento de débitos [Almeida (2004)].

A segunda *Configuration*, denominada CDC, é baseada na máquina virtual Java convencional, definindo um ambiente rico de recursos semelhantes aos encontrados em um sistema *desktop*. Está voltada para dispositivos dotados de maior capacidade computacional com no mínimo 2 MB de memória disponível [Rischpater (2001)]. Por exemplo, *set-top boxes* de TVs a cabo, sistemas automotivos e outras plataformas que possuam pelo menos alguns *megabytes* de memória disponível.

Profiles são conjuntos de APIs que suplementam *Configurations*, fornecendo funcionalidade para um determinado tipo de dispositivo ou mercado vertical. A Sun introduziu o conceito de *Profile* na plataforma J2ME para tratar da ampla variação de recursos e dar mais flexibilidade à medida que a tecnologia se modifica [Topley (2002)].

Um *Profile* atende às demandas específicas de certa família de dispositivos. São mais específicos que as *Configurations*, apesar de serem baseados nelas. Enquanto uma *Configuration* visa aparelhos que possuem recursos de *hardware* semelhantes, um *Profile* é definido para dispositivos que executam tarefas semelhantes [Luz *et al.* (2005)]. Ao contrário de uma *Configuration*, *Profile* inclue bibliotecas mais específicas e vários *Profiles* podem ser suportados pelo mesmo dispositivo. Além disso, as classes que fazem parte de um *Profile* se estendem àquelas definidas para uma *Configuration*.

O MIDP (*Mobile Information Device Profile*) foi o primeiro *Profile* definido para a plataforma J2ME tendo sido lançado em novembro de 1999 [Pereira *et al.* (2004)]. Esse *Profile* foi implementado sobre a *Configuration* CLDC e define APIs para componentes, entrada e tratamento de eventos de interface com o usuário, armazenamento persistente e interligação em rede e cronômetros, levando em consideração as limitações de tela e memória dos dispositivos móveis [Luz *et al.* (2005)].

O *Personal Basis* e o *Personal Profile* são Perfis desenvolvidos para a *Configuration* CDC utilizados em dispositivos com suporte gráfico e alta capacidade [Muchow (2004)].

As APIs opcionais são funções adicionais específicas que não serão encontradas nos dispositivos de uma determinada *Configuration* ou *Profile* [Almeida (2004)]. As APIs opcionais mais conhecidas são a *WMA (Wireless Messagung API*) que permite aos aplicativos J2ME manipular mensagens *SMS* e a *MMAPI (Mobile Media API)* que adiciona controle de mídia aos programas J2ME nos dispositivos que o permitem.

Segundo Gomes (2007), a plataforma J2ME hoje se encontra em intensa evolução. Sobre as duas *Configurations* básicas, vários *Profiles* vêm sendo definidos. Tendo sido projetado para dispositivos limitados em termos de capacidade computacional, o *Profile* para CLDC possui diversas restrições que não estão presentes em outras versões da linguagem Java. Dentre estas limitações, cita-se a não existência do tipo ponto flutuante utilizado para representar números reais. Outra limitação importante é a ausência de suporte aos mecanismos de reflexividade presentes em Java, os quais permitem que informações sobre a estrutura interna de um programa sejam conhecidas enquanto ele está sendo executado, por exemplo, o tipo dinâmico de um objeto. A Figura 3.4 ilustra a composição da plataforma J2ME atualmente.

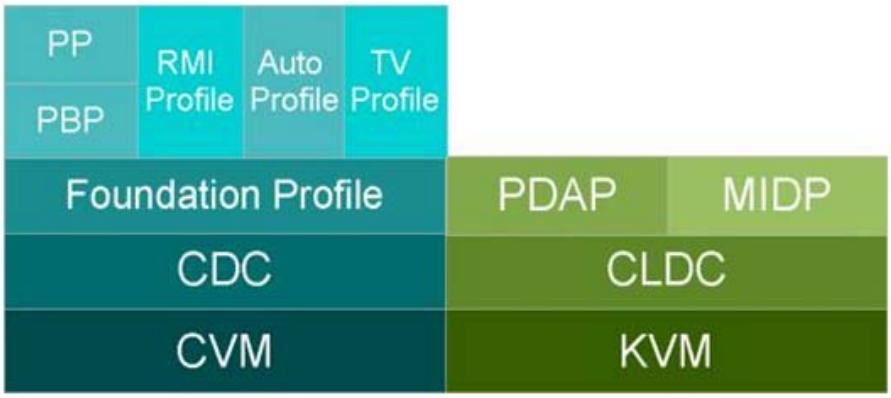

Figura 3.4 – J2ME Hoje [Fonte: Gomes(2007)]

3.4. Vantagens do J2ME

O uso de J2ME no desenvolvimento de aplicações para dispositivos móveis significa ganhar as vantagens que a tecnologia traz consigo, dentre elas podem-se citar [Gomes (2007)]:

- Dinamismo: novas aplicações podem ter a opção de *download* através da rede e instaladas no dispositivo a qualquer tempo;
- Segurança: *class verification*, forte tipagem e *garbage colection*, entre outros, garantem a proteção das informações carregadas pelo dispositivo. Dados de uma aplicação não são acessíveis por outras aplicações;
- Portabilidade: capacidade de ser facilmente transportável. Aplicações podem ser portadas entre dispositivos de diferentes fabricantes e de diferentes tipos;
- Orientação a Objetos: alto nível de abstração do código, modularização e reusabilidade;

- Comunidade: estima-se que existam cerca de 2,5 milhões de desenvolvedores Java no mundo ao mesmo tempo em que tem se tornado a linguagem de programação mais popular ensinada em cursos de atualização e universidades;
- Confiabilidade: não é tolerado *reboots* ou *crashs* em dispositivos embarcados. Graças aos mecanismos de proteção e gerência de memória, Java atinge esta demanda [Topley (2002)].

3.5. Considerações Finais

J2ME é uma maneira simples e econômica de trazer para os usuários o acesso a seus sistemas corporativos e informações pessoais além do entretenimento. A edição J2ME não é só uma tecnologia de futuro, mas uma tecnologia atual, é uma realidade. Ela é a base para aplicações móveis que visam oferecer ao usuário liberdade e individualização do seu aparelho móvel, além de fornecer aplicações atualizadas por milhões de desenvolvedores [Caldeira (2005)]. É uma tecnologia que se encontra presente no dia a dia de muitos usuários e vai se tornar cada vez mais onipresente.

Verifica-se nas empresas de telefonia móvel que vem ocorrendo uma redução na receita originada por serviços baseados em voz. Em função deste acontecimento, as operadoras vêm oferecendo serviços baseados em dados para aumentar a receita média por assinante e, para isso, vem utilizando a cada dia mais a tecnologia J2ME [Luz *et al.* (2005)].

4. *WEBSERVICES*

4.1. Considerações Iniciais

Nos últimos anos, a *World Wide Web*, ou simplesmente *Web*, vem disponibilizando comunicação, entretenimento, comércio, serviços e acesso a documentos aos usuários.

Verifica-se que atualmente muitas empresas vêm permitindo às pessoas, internas e externas à empresa, o acesso aos seus serviços via *Internet*, desde serviços de compra e consulta a preços à solicitação de serviços. A utilização da *Web* como meio de integração entre determinadas aplicações de uma empresa, além de permitir ao cliente ou ao funcionário uma melhor comodidade, gera grande benefício e lucro para a empresa que a utiliza, pois é um meio mais rápido, barato e eficiente com relação aos dados necessários na solicitação de algum serviço disponível pela empresa.

No emprego da *Web* como forma de integração entre aplicações diferentes, é necessária a escolha de uma linguagem em comum. Ou seja, é necessária a utilização de uma solução na integração de sistemas e na comunicação entre aplicações diferentes, desenvolvidas em plataformas heterogêneas. Com base nessa necessidade, surge o conceito de *WebServices* que, conforme Pamplona (2006), é uma tecnologia que permite novas aplicações interajam com aquelas existentes e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis, ou seja, são compostos por componentes que gerenciam a parte de negócio das aplicações a ele envolvidas.

Segundo Urresti (2002), *WebService* é um sistema de *software* desenvolvido para suportar comunicação de aplicações entre máquinas. Além disso, o grande objetivo dos *WebServices* é permitir que a comunicação seja possível entre máquinas heterogêneas, promovendo a interoperabilidade entre plataformas distintas através do uso de normas baseadas em XML (*eXtensible Markup Language*). Ou seja, *WebServices* são componentes que permitem às aplicações enviarem e receberem dados onde cada uma pode ter a sua própria linguagem, traduzida em uma linguagem universal: a XML.

Os *WebServices* permitem que a integração de sistemas seja realizada de maneira compreensível, reutilizável e padronizada. É uma tentativa de organizar um cenário cercado por uma grande variedade de aplicativos, fornecedores e plataformas.

Para as empresas, *WebServices* podem trazer agilidade para os processos e eficiência na comunicação entre cadeias de produção ou de logística. Toda e qualquer comunicação entre sistemas passa a ser dinâmica e, principalmente, segura, pois não há intervenção humana.

A simplificação do processo de integração de aplicações é o grande objetivo de *WebServices*. Rompendo barreiras entre sistemas, arquiteturas, *hardware* e outros aspectos, *WebServices* entregam promessas antigas da perspectiva de integração. As facilidades técnicas apresentadas pelos *WebServices* [Urresti (2002)]:

- O uso de XML: é o formato pressuposto para envio e recebimento de informações. XML facilita a troca e a transformação de informação;
- Não necessitam de *brokers*: *WebServices* não necessitam do uso de *brokers* (servidor especial de tratamento de requisições a objetos remotos), eliminando um dos fatores de interoperabilidade;
- Transporte da informação via HTTP: *WebServices* usam tipicamente o protocolo HTTP (*Hyper Text Transfer Protocol*) para transportar seus dados. A grande limitação é a sua baixa eficiência em termos de robustez. Mas, nesses casos, é possível usar outros protocolos de transporte;
- Independência de plataforma e de linguagem de desenvolvimento: não há requisitos de plataforma, permitindo flexibilidade no desenvolvimento e simplificação do processo de integração entre ambientes heterogêneos.

4.2. Arquitetura

Segundo Amaral (2002), o que difere *WebServices* das demais arquiteturas são suas características de implementação: os protocolos que compõem a arquitetura baseiam-se principalmente em padrões aceitos e adotados pelo mercado – o protocolo HTTP e o padrão de representação de dados XML.

As bases para a construção de um *WebService* são os padrões XML e SOAP (*Simple Object Access Protocol*). O transporte dos dados é realizado, normalmente, via protocolo HTTP (o padrão não determina o protocolo de transporte). Os dados são transferidos no formato XML, encapsulados pelo protocolo SOAP.

O uso do protocolo HTTP para a troca de mensagens faz com que qualquer *WebServer* possa, sem dificuldades, atender requisições por *WebServices* como se fosse um *broker* ou registros de objetos [Freire (2002)].

Para a representação e a estruturação dos dados nas mensagens recebidas/enviadas é utilizada XML. XML é utilizada para gerar linguagens de marcação para necessidades especiais. É um subtipo de SGML (*Standard Generalized Markup Language*) capaz de descrever diversos tipos de dados. O objetivo principal de XML é a facilidade de compartilhamento de informações através da *Internet*.

XML é considerada uma boa linguagem para a criação de documentos com dados organizados de forma hierárquica, como se vê freqüentemente em documentos de texto formatados, imagens vetoriais ou bancos de dados [Amaral (2002)].

As chamadas às operações, incluindo os parâmetros de entrada/saída, são codificadas no protocolo SOAP. As mensagens SOAP são documentos XML que aderem a uma especificação fornecida pelo órgão W3C (*World Wide Web Consortium*). O SOAP é um protocolo para intercâmbio de mensagens entre programas de computador usados na criação de *WebServices*. Geralmente, servidores SOAP são implementados utilizando-se servidores HTTP pré-existentes, embora isto não seja uma restrição para funcionamento do protocolo.

O protocolo que permite as chamadas entre objetos escritos em linguagens distintas é o SOAP, um protocolo simples, que define chamadas de procedimentos e respostas na forma de elementos XML. O SOAP baseia-se na idéia de um mecanismo de RPC (*Remote Procedure Call*) em XML [Urresti (2002)].

As mensagens trocadas entre as aplicações usam o formato SOAP e, deste modo, tentam normalizar o tradicional RPC. Assim, é possível que duas aplicações, que usem tecnologias diferentes (Java e .Net), comuniquem-se sem problemas [Sun (2006a)]. A desvantagem de usar esta tecnologia é o uso de um formato de texto (XML) o qual faz com que as mensagens transmitidas sejam maiores e, por isso, use uma maior largura de banda.

Segundo Pamplona (2006), o protocolo SOAP é a base principal da arquitetura de *WebServices*, mas não é o único protocolo da arquitetura. À medida que novas implementações foram desenvolvidas, surgiu a necessidade de novos protocolos. Além de

um protocolo de troca de mensagens, foram divulgados dois outros padrões – um protocolo de localização de objetos e um esquema de representação de dados.

Os serviços (operações, mensagens, parâmetros) são descritos usando a linguagem WSDL (*Web Services Definition Language*). O processo de publicação/pesquisa/descoberta de *WebServices* utiliza o protocolo UDDI (*Universal Description, Discovery and Integration*) [Amaral (2002)].

O padrão WSDL foi o último dos três protocolos que compõem a arquitetura e foi lançado algum tempo depois do UDDI. Para descrever os objetos, os parâmetros e os dados de forma universal, foi criada uma gramática descritiva de objetos e serviços baseada em XML [Freire (2002)].

O protocolo UDDI, segundo Freire (2002), foi proposto para permitir o armazenamento e a localização de *WebServices*. Este protocolo dita como *WebServices* podem ser registrados e localizados na rede.

A adoção de padrões abertos e universalmente aceitos tem como objetivo garantir dois pontos que foram esquecidos nos padrões de distribuição de objetos anteriormente propostos: a interoperabilidade e a manutenção de código legado [Amaral (2002)].

Conforme Freire (2002), a interoperabilidade é um dos principais pontos de destaque dos *WebServices*: na grande disputa por mercado entre a *Sun* e a *Microsoft*, os *WebServices* surgiram como uma ponte de integração entre serviços propostos nas duas arquiteturas (.Net e J2EE).

O aproveitamento do código legado escrito, por exemplo, em Java, e a sua transformação em uma funcionalidade pronta para ser utilizada por um outro programa escrito em qualquer linguagem é o mais estimulante na arquitetura *WebService*. Como os componentes acessados utilizando os protocolos da arquitetura são descritos usando a linguagem XML, o cliente pode ser escrito em uma linguagem diferente daquela usada pelo provedor de serviços – a implementação dos protocolos da arquitetura *WebServices* nas diversas plataformas de programação se encarrega de promover esta conversão transparente dos dados [Pamplona (2006)].

A criação de interfaces gráficas para os usuários não faz parte do conceito de *WebService*, deixando esta parte para outras empresas ou pessoas desenvolverem.

Conforme, Freire (2002), *WebServices* disponibilizam serviços somente para desenvolvedores que chamam métodos usando XML.

4.3. Futuro

Acredita-se que no futuro, as empresas irão listar seus *WebServices* em diretórios públicos UDDI, de onde poderão ser vendidos como serviços para outras empresas, instituições ou usuários comuns.

WebServices vêm passando por inúmeras transformações desde a sua primeira fase até a atual e com grandes idéias para futuras modificações.

- Na primeira fase, *WebServices* Simples, o desenvolvimento interno é concentrado em uma organização com o objetivo de conseguir maior interoperabilidade;
- Na segunda fase, *WebServices* EAI (*Enterprise Application Integration*), o desenvolvimento é seletivo e não agregatório com parceiros de negócios externos devidamente selecionados e implementação de registros privados;
- Na terceira fase, *WebServices* de Negócio, o desenvolvimento é mais vasto, dinâmico e agregatório com parceiros de negócios externos e implementação de registros públicos.

Apesar de pouco tempo de existência e da grande quantidade de problemas potenciais ainda existentes, especialmente no que diz respeito à segurança, *WebServices* não podem ser ignorados. Mais que apenas um outro padrão de interoperabilidade, *WebService* tem tudo para se tornar à base da revolução na *Internet* [Pamplona (2006)].

4.4. Considerações Finais

WebServices são componentes de *software* que podem ser invocados programaticamente por outros componentes através da *Internet*.

A tecnologia *WebService* visa a aumentar a interoperabilidade entre componentes e diminuir o custo de integração entre aplicações. Embora seja uma tecnologia recente, ela está sempre sendo inovada pelas principais empresas de *software* do mundo.

Analisando as inúmeras razões para usar *WebServices*, podem-se destacar [Urresti (2002)]:

- Interoperabilidade – permite ligações entre componentes de *software* que podem estar em diferentes empresas e podem residir em diferentes infra-estruturas;

- Economia – permite a reciclagem de componentes, não é preciso instalação nem uma integração muito forte deles;
- Automatização – não é necessária intervenção humana mesmo em transações muito complexas;
- Acessibilidade – sistemas legados e aplicações internas são expostos e podem ser cedidas na *Web*;
- Disponibilidade – serviços em qualquer dispositivo, em qualquer lugar, a qualquer hora;
- Escalabilidade – não existem limites no âmbito das aplicações nem na quantidade de aplicações heterogêneas.

5. SISTEMA SisBDR

5.1. Considerações Iniciais

O avanço da tecnologia tem estimulado o desenvolvimento de aplicações mais robustas. A mobilidade vem sendo uma das tecnologias mais bem sucedidas nos últimos tempos e, com isso, o desenvolvimento de aplicações para tais dispositivos torna-se um dos temas de maior investimentos e estudos na atualidade.

Com base nesta recente tecnologia, o objetivo do trabalho é o desenvolvimento de uma aplicação para um dispositivo móvel que se conecta em um servidor de banco de dados, após a realização de uma das consultas, simula a perda de conexão e, automaticamente, se conecta em outro servidor para que a outra consulta seja realizada com êxito.

Nas consultas a serem realizadas pelo sistema, foi utilizado um caso auxiliar que será mais bem explicado neste capítulo, as consultas se relacionam a hospedagens, visando a um melhor conforto para os usuários na procura por lugares desejados para se hospedarem.

Conforme descrito no Capítulo 3, duas configurações são possíveis: CLCD e CDC. Na implementação do sistema SisBDR, foi utilizada a configuração CLCD-1.1 e o *Profile* MIDP-2.0, pois a maior parte dos dispositivos móveis utilizam tal configuração e este *Profile* é o único que suporta a configuração.

O processamento de consultas é realizado no servidor, que também é responsável pelo *WebService*.

5.2. Arquitetura

O SisBDR foi desenvolvido sob a arquitetura cliente/servidor e pode ser visto como uma estrutura repartida em módulos de funcionamento no cliente e no servidor.

A classe que trata o interfaceamento com o usuário através de dispositivos móveis faz requisições ao sistema servidor, podendo direcionar o acesso a bases de dados distintas e as consultas desejadas localizada no cliente.

A parte de negócios tem o papel de abranger o processamento e o funcionamento do sistema, trata da parte lógica dos dados. Dessa forma, parte dela fica por conta do servidor,

para as consultas e o retorno dos dados requeridos, enquanto o cliente realiza processamento antes da exibição dos dados propriamente ditos.

O acesso a dados é mantido pelo servidor e pode ser dito como o conjunto de aplicações ou elementos que são utilizados para armazenamento de dados essenciais ao SisBDR. Assim, como elemento principal, tem-se o MySQL, que possui como benefícios visíveis o fato de ser livre e manter a robustez necessária para o gerenciamento das informações.

O acesso do cliente aos dados mantidos no SisBDR é realizado de forma a dar flexibilidade quanto à decisão pelo banco de dados a ser acessado. Assim, o usuário pode fazer a escolha do banco de dados e o acesso às informações requeridas será direcionado à este banco de dados. Durante a realização das consultas, o sistema se encarrega de simular uma perda de conexão com o servidor de banco de dados, em seguida, se conecta em um outro servidor e retorna os resultados das consultas com êxito.

Dois bancos de dados que contêm informações similares são mantidas (redundância de informações nos servidores de banco de dados). Entretanto, o acesso é feito a cada uma delas separadamente, dependendo da escolha do usuário.

No processamento das consultas, quando o sistema se desconectar e, automaticamente, se conectar no outro banco de dados, o sistema se encarrega de passar as consultas ainda não realizadas para o novo banco de dados conectado.

5.3. Funcionamento do Sistema

O sistema possibilita ao usuário realizar consultas relacionadas a hospedagens de determinado estado através de um dispositivo móvel.

Na tela principal do sistema, são apresentadas três perguntas ao usuário, nas quais deve ser selecionada a resposta desejada para que as consultas possam ser realizadas:

- O usuário deve selecionar o local em que ele se encontra, na simulação SP ou MG, para que o SisBDR possa se conectar no servidor de banco de dados mais adequado;
- O SisBDR permite ao usuário escolher o local da hospedagem para a consulta. Uma listagem de todas as hospedagens localizadas no local selecionado pelo usuário é apresentada pelo SisBDR;

- As hospedagens cadastradas nos servidores são do tipo Hotel e Pousada, o SisBDR permite ao usuário escolher o tipo desejado para a consulta e são apresentadas todas as hospedagens do tipo escolhido pelo usuário.

Após o processamento, são mostrados os resultados das consultas, apresentando ao usuário o nome, o telefone, a cidade, o estado, e o tipo da hospedagem. A arquitetura do sistema é apresentada na Figura 5.1.

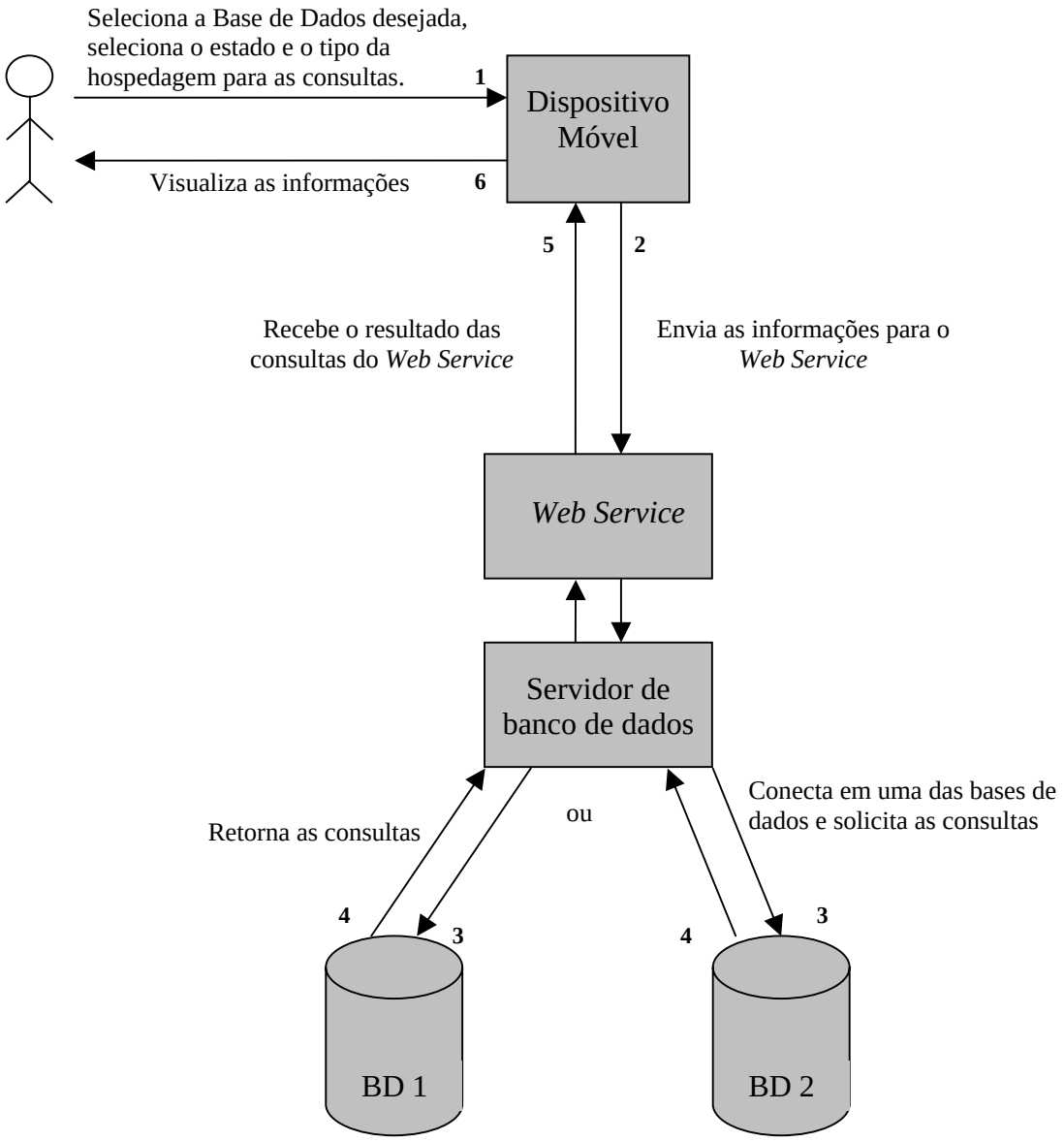

Figura 5.1 – Arquitetura do SisBDR

Nesta arquitetura, o usuário realiza os seguintes passos:

1. O usuário seleciona o banco de dados desejado, seleciona a localidade desejada para a consulta e o tipo da hospedagem para a consulta, através de um dispositivo móvel;

2. O nome do banco de dados desejado, a localidade e o tipo da hospedagem são enviados como parâmetro ao *WebService*;
3. *WebService* é encarregado de passar os parâmetros para o servidor de banco de dados;
4. Após feita a conexão com o servidor de banco de dados, a primeira consulta é realizada e depois de ser retornada para o *WebService*, uma perda de conexão é simulada. Logo em seguida, *WebService* se conecta no outro servidor e a outra consulta que ainda não foi realizada é solicitada, então o servidor retornar o resultado ao *WebService*;
5. *WebService*, por sua vez, encarrega-se de repassar os resultados para o celular. Após a recepção dos dados pelo celular, são exibidas na tela as informações sobre as hospedagens localizadas no estado selecionado e as hospedagens que são do tipo escolhido pelo usuário;
6. O usuário visualiza as informações encontradas.

5.4. Modelagem do Sistema

A Figura 5.2 apresenta o Diagrama de Caso de Uso do SisBDR. O ator usuário poderá consultar por hospedagens.

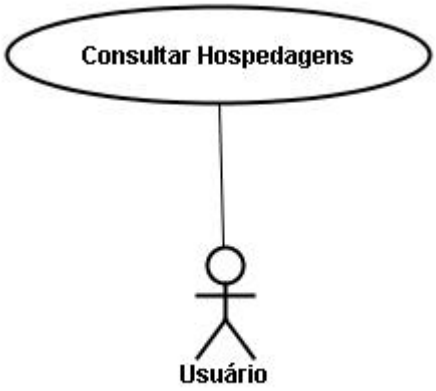

Figura 5.2 – Diagrama de Caso de Uso do SisBDR

5.4.1. *WebService*

Muitas ferramentas tornam mais rápida e fácil a implementação de *WebServices*. Durante a criação do *WebService*, são passadas as operações desejadas e, a partir das ferramentas disponíveis, é gerado o código necessário. É preciso somente o desenvolvimento da parte do servidor de banco de dados, ou seja, os métodos de conexão e de consulta.

O Diagrama de Classes do módulo *WebService* é mostrado na Figura 5.3. As classes *WebToMobileServlet* e *WebServiceSisBDR* é o *WebService* mencionado. A classe *WebToMobileServlet* é o ponto de comunicação entre *WebService* e o cliente móvel. A classe que acessa diretamente a base de dados é representada pela classe *Conexao* e a classe *SisBDR* que passa para a classe *Conexão* as consultas desejadas.

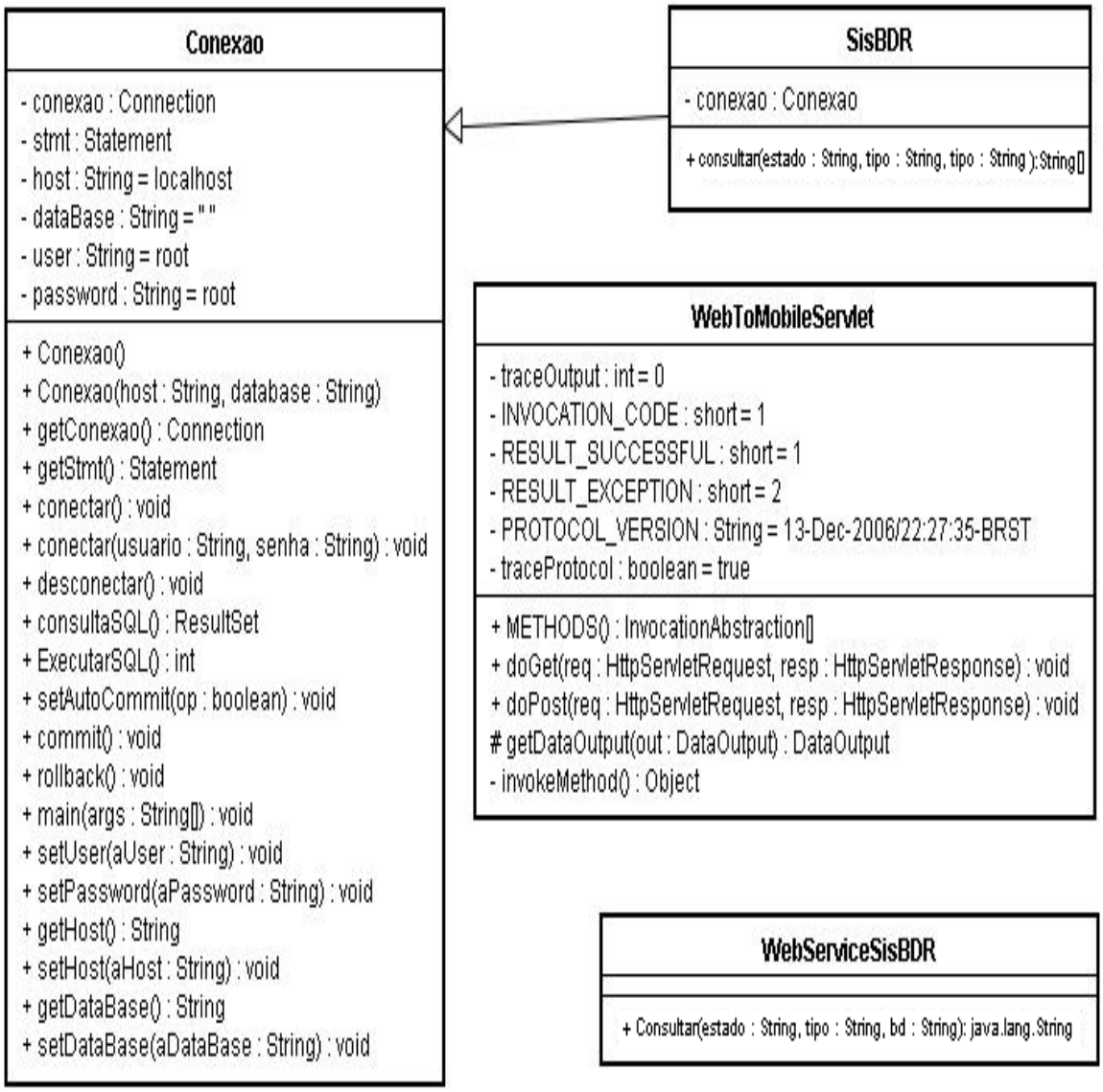

Figura 5.3 – Diagrama de Classe do módulo Servidor

O Diagrama de Seqüência permite a visualização das seqüências de ações quando o cliente faz a solicitação de consultas ao *WebService*. A classe *WebtoMobileServlet* recebe as requisições do dispositivo móvel e envia para a classe *WebServiceSisBDR*. A classe *WebServiceSisBDR* se encarrega de enviar as requisições para a classe *SisBDR* que contém os métodos necessários para a realização das consultas e acessa os servidores de banco de dados para dar resposta as requisições (Figura 5.4).

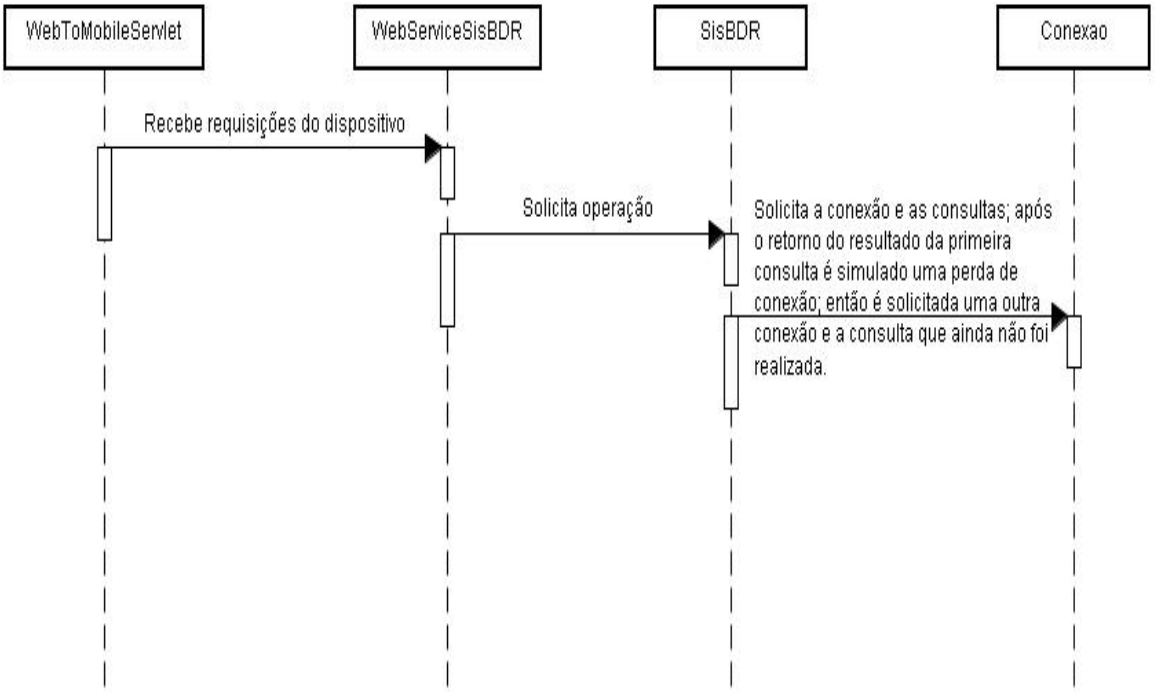

Figura 5.4 – Diagrama de Seqüência

5.4.2. Cliente

No desenvolvimento do cliente do *WebService*, foram utilizadas algumas ferramentas que contribuíram no processo de desenvolvimento, sendo modificada apenas a parte de interface com o usuário.

A Figura 5.5 ilustra o *Flow Designer* do projeto que foi gerado pelo *NetBeans*. Desde a seleção da operação às telas que imprimem os resultados das consultas, conforme descrito a seguir:

1. o usuário seleciona a aplicação desejada;
2. as operações disponíveis na aplicação são apresentadas nesta tela, onde o usuário deve selecionar a operação a ser realizada, no caso do SisBDR, Consultar Hospedagem. Nesta tela, é possível alterar a URL (*Uniform Resource Locator*) do servidor do qual a aplicação móvel é cliente;
3. as consultas disponíveis e os servidores de banco de dados disponíveis são apresentadas nesta tela. O usuário deve selecionar as opções desejadas;
4. se o usuário deseja alterar a URL do servidor, uma tela é apresentada contendo o endereço do servidor utilizado, de forma que o usuário possa alterá-lo;
5. na tela, é mostrada ao usuário que será solicitado a conexão com o *WebService*;

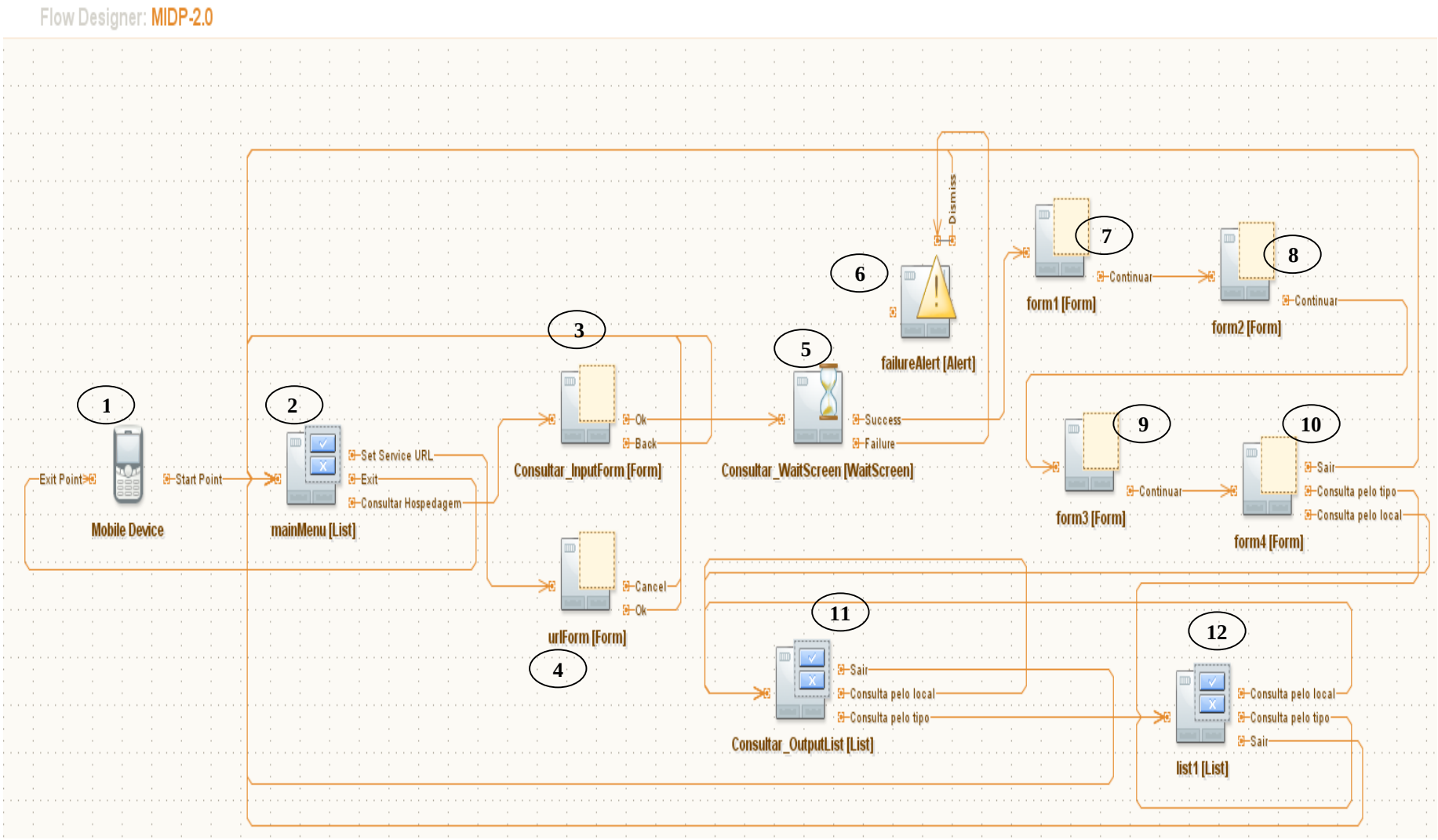

Figura 5.5 – *Flow Desinger* do projeto

6. se não for possível ser feita a conexão com o *WebService*, é exibida uma tela de erro ao usuário e o SisBDR;
7. se for possível a conexão com o *WebService*, as consultas são solicitadas. Após a realização da primeira consulta, é simulada perda de conexão com o servidor de banco de dados e é informado ao usuário que ocorreu uma perda de conexão. Em seguida, é solicitada a conexão com outro servidor de banco de dados para realizar a segunda consulta. Nesta tela, é mostrado ao usuário que ele mudou de área;
8. perda de conexão com o servidor de banco de dados;
9. estabelecimento da conexão em outro servidor de banco de dados;
10. informa ao usuário que as consultas foram realizadas com sucesso e permite ao usuário, através de um *menu*, selecionar as consultas para que os resultados possam ser apresentados;
11. e 12. apresenta os resultados da consulta seleciona e permite ao usuário selecionar a outra consulta ou então sair do sistema.

A Figura 5.6 mostra o Diagrama de Classes do módulo Cliente que é composto por apenas duas classes:

- *WebToMobileClientMIDlet* – classe que trata a interface com o usuário do celular;
- *WebToMobileClient* – comunica diretamente com a classe *WebToMobileServlet* do módulo *WebService*.

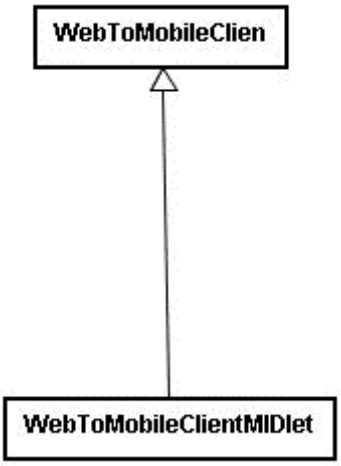

Figura 5.6 – Diagrama de Classes do módulo Cliente

A Figura 5.7 mostra o Diagrama de Seqüência para o módulo Cliente. No diagrama, a classe *WebToMobileClientMIDlet* envia o comando do usuário para a classe *WebToMobileClient*, que solicita uma resposta do *WebService*.

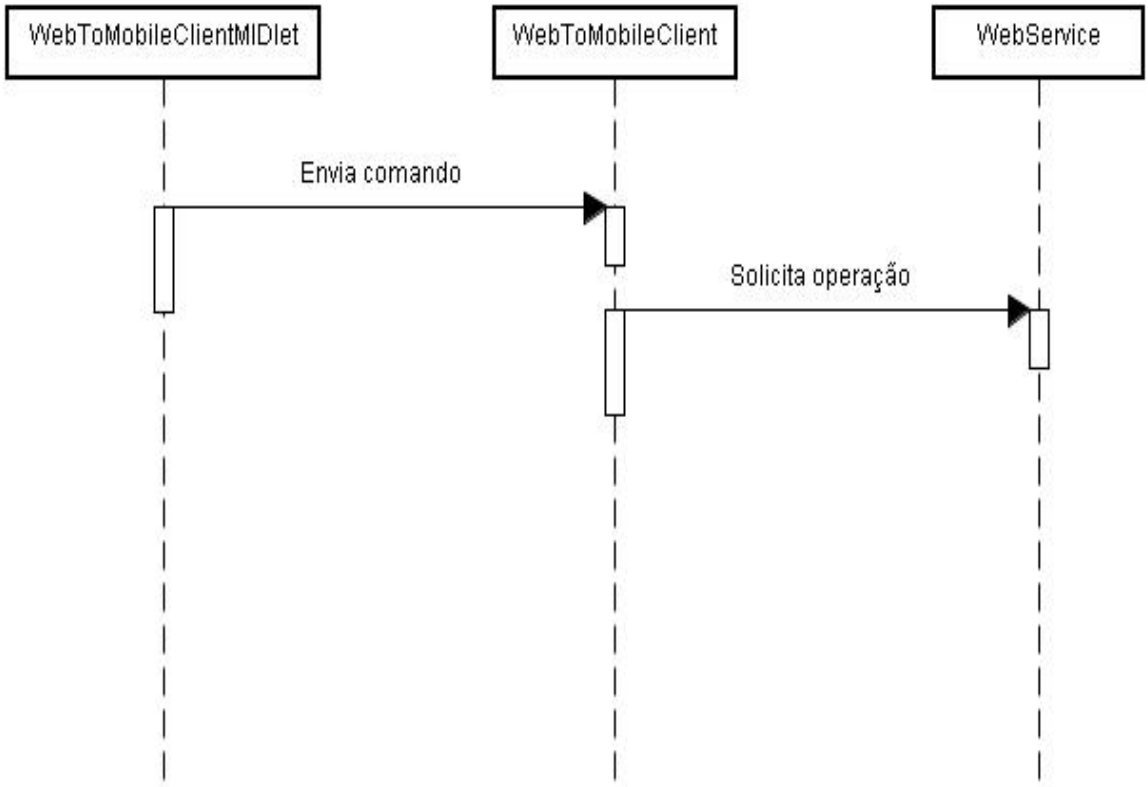

Figura 5.7 – Diagrama de Seqüência

5.5. Contexto do SisBDR

O turismo assumiu nos últimos anos o papel de maior atividade civil do mundo em termos de geração de renda e emprego. É com base nesses dados que os investimentos a locais para hospedagem crescem a cada dia.

O Plano de Aceleração do Crescimento (PAC) propôs um investimento de R$ 58,3 bilhões na infra-estrutura de logística do país até 2010, o que beneficiará diretamente o setor de Turismo [CET/UNB (2007)].

Atualmente, a consulta por hospedagens pode ser feita pela *Internet* ou por outros meios de comunicação como jornais, revistas e empresas voltadas para o turismo. No entanto, percebe-se que os meios de consulta nem sempre são eficientes. Esses meios de consulta em pequenos e em grandes centros não conseguem atingir a população. Com base nessas dificuldades, há a necessidade de criar meios mais eficientes, possibilitando maior divulgação das hospedagens, acesso seguro e melhor conforto aos usuários.

O caso a ser desenvolvido foi baseado na necessidade da manutenção do cadastro de hospedagens, trata-se de um sistema auxiliar no objetivo principal do trabalho que é o sistema SisBDR. Ou seja, o Sistema de Cadastro de Hospedagens visa a manter o cadastro

de hospedagens nos servidores de banco de dados e o usuário ao utilizar SisBDR através de um dispositivo móvel poderá fazer consultas as hospedagens.

Na manutenção do cadastro das hospedagens, é necessário informar o código, o nome, o telefone, a cidade, o estado e o tipo da hospedagem.

No desenvolvimento do sistema, foram adotadas as seguintes tecnologias: i) linguagem de programação Java; e ii) MySQL.

5.6. Arquitetura do Caso de Uso

O código do sistema foi dividido em camadas para melhor organização e maior facilidade na sua manutenção.

As classes que tratam a interface do sistema são responsáveis por exibirem as telas existentes conforme as solicitações do usuário e fazem parte da camada de apresentação e se encontram no pacote *interface*. A parte de negócios tem o papel de abranger o processamento e o funcionamento do sistema. As classes estão organizadas na camada de negócios, no pacote *manager*. O acesso aos dados é responsabilidade das classes localizadas no pacote *dao*, essas classes contêm métodos de manutenção dos dados.

5.7. Modelagem do Caso de Uso

A Figura 5.8 apresenta o Diagrama de Caso de Uso. O ator usuário pode realizar as operações de manutenção dos dados das hospedagens.

Figura 5.8 – Diagrama de Caso de Uso

O código-fonte, como descrito no tópico Arquitetura, foi dividido em camadas. O Diagrama de Classe da Figura 5.9 ilustra as classes utilizadas e a divisão em camadas das classes.

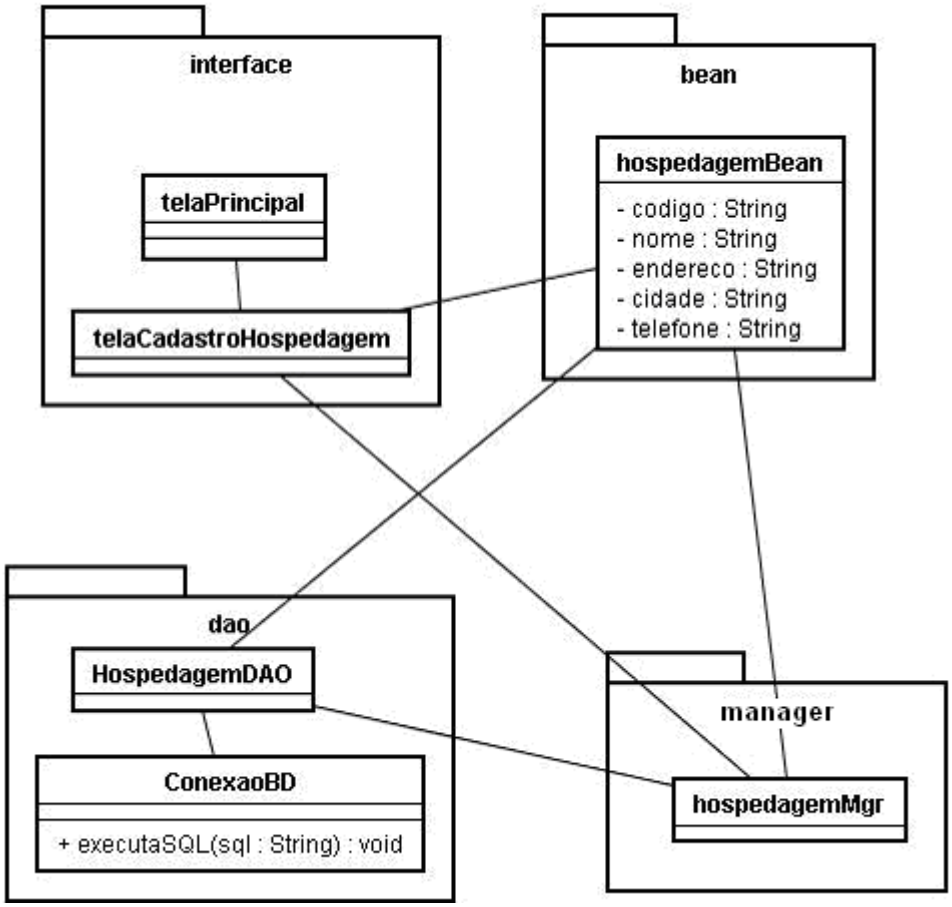

Figura 5.9 – Diagrama de Classes

5.8. Funcionamento do Caso de Uso

O sistema possibilita ao usuário fazer inclusões, consultas, alterações e exclusões de hospedagens através de uma interface gráfica. O usuário pode realizar essas operações através do código da hospedagem.

O programa, ao ser executado, tentará estabelecer diretamente a conexão com banco de dados pré-definido. Caso não seja possível estabelecer a conexão, uma mensagem de erro aparece e as operações disponíveis no sistema não são habilitadas.

Caso a conexão com o banco de dados seja realizada com sucesso, a tela principal é apresentada contendo o *menu* Hospedagem que contém as seguintes opções: Incluir, Consultar, Alterar, Excluir e Sair (Figura 5.10).

Na operação de inserção, o usuário deve preencher um formulário contendo código, nome, telefone, endereço, cidade e telefone da hospedagem a ser inserida no banco de dados. Após o usuário preencher os campos e confirmar a operação, o sistema inclui estas

informações no banco de dados. Uma mensagem é apresentada ao usuário, informando se a operação foi realizada com sucesso ou se não foi possível realizar a inserção dos dados.

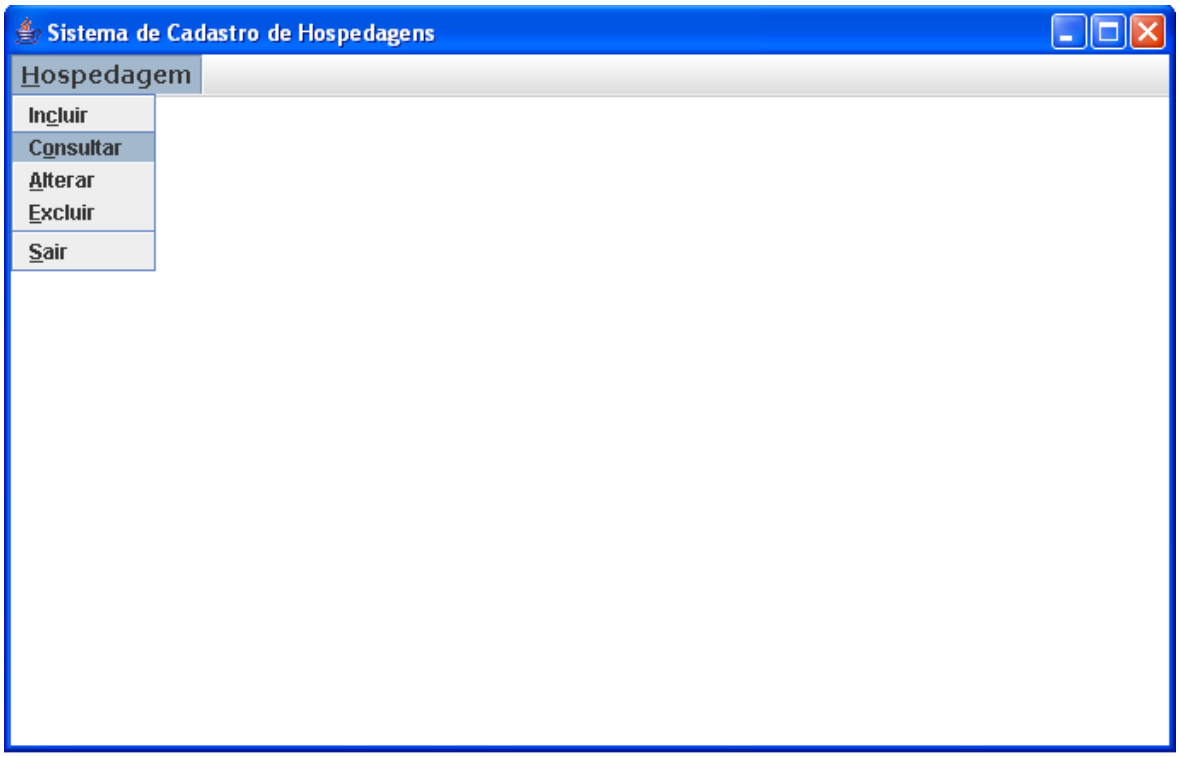

Figura 5.10 – Tela Principal do Sistema de Cadastro de Hospedagem

No momento da inserção, a aplicação faz uma consulta ao banco de dados através do código para verificar se a hospedagem que deseja cadastrar existe. Se existir, uma mensagem é mostrada e a inserção não ocorre; caso contrário, os dados são inseridos.

Na alteração dos dados, o usuário apenas pode alterar dados, exceto o código de hospedagens que existam no banco de dados. Assim que o usuário modifica os dados e confirma a operação, é exibida uma mensagem informando se a operação foi concluída.

Na consulta, o usuário deve informar o código da hospedagem. Se o código constar no banco de dados, seus dados são impressos na tela; caso contrário, uma mensagem informa ao usuário que a hospedagem não se encontra cadastrada no banco de dados (Figura 5.11).

5.9. Demonstração do SisBDR

Uma demonstração do SisBDR é apresentada nesta seção. Para a demonstração, foi suposto que o usuário se encontre em SP (São Paulo) e solicite uma consulta por hospedagens de MG (Minas Gerais) e do tipo Hotel.

Para execução dos testes, foram utilizados um simulador do *NetBeans* e um microcomputador com o *Sun Java System Application Server PE* para a execução do *WebService*.

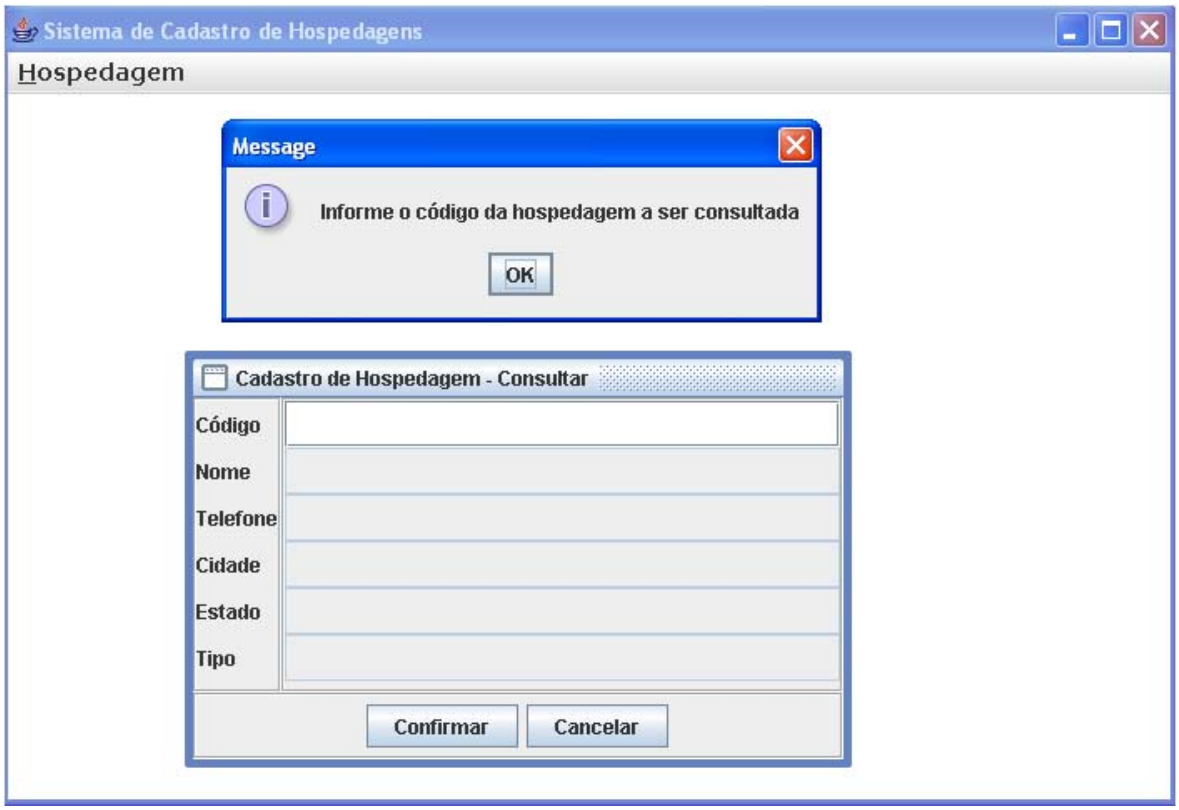

Figura 5.11 – Tela de consulta a hospedagem

A Figura 5.12 apresenta o simulador com a máquina virtual e o SisBDR instalados. A Figura 5.13 apresenta a tela inicial do sistema e a operação de consulta do sistema. A URL do *WebService* pode ser alterada, caso ele esteja hospedado em um servidor diferente (Figura 5.14). A Figura 5.15 ilustra a tela onde são selecionados os dados para a consulta, o local da em que se encontra o dispositivo, o local da hospedagem e o tipo de hospedagem.

Depois de selecionadas as opções, são solicitadas as consultas. O dispositivo se comunica com o servidor e este se encarrega da comunicação com os servidores de banco de dados (Figura 5.16). Após o servidor retornar o resultado da primeira consulta, uma perda de conexão é simulada e é solicitada a conexão com outro servidor. Como a princípio foi escolhida a localidade SP, então será solicitada a conexão ao servidor na localidade MG para que a consulta pendente seja realizada. Cabe ressaltar que as informações estão replicadas nos dois servidores de banco de dados bases. A Figura 5.17 mostra a tela que o SisBDR exibe ao usuário informando que ocorreu a perda de conexão.

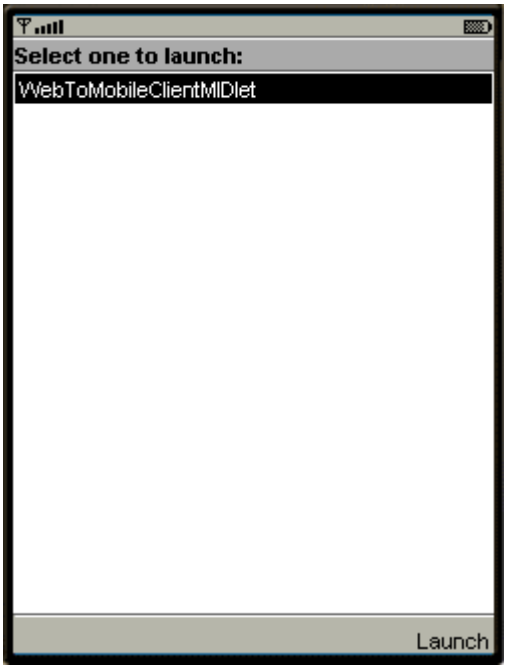

Figura 5.12 – Simulador com o SisBDR instalado

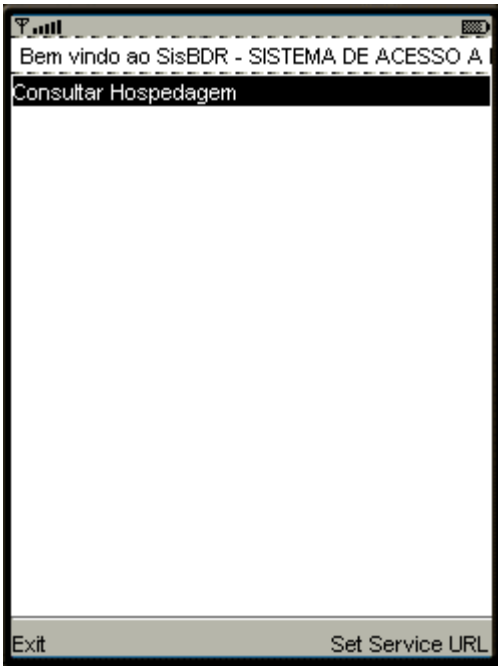

Figura 5.13 – Tela inicial do SisBDR

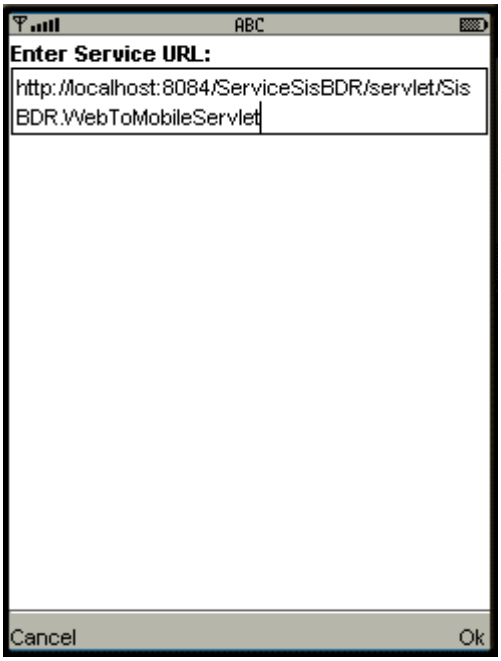

Figura 5.14 – Alterar a *URL* do *WebService*

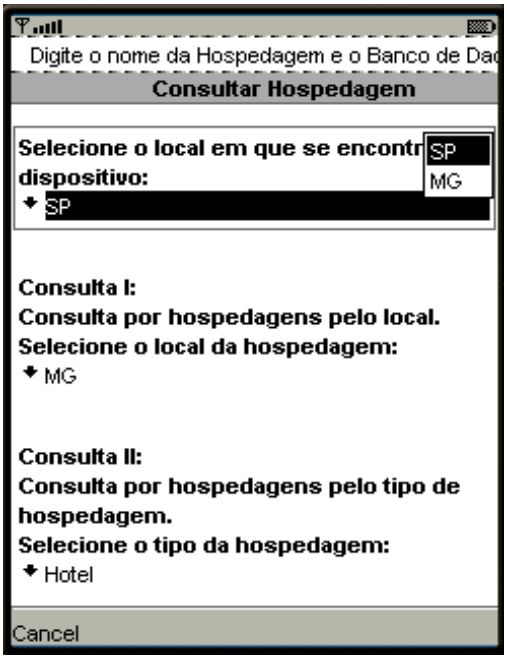

Figura 5.15 – Tela opções para a consulta

Após a conexão com o outro servidor estiver estabelecida, é solicitada a consulta que ainda não foi realizada. Com as consultas realizadas, os resultados são mostrados na tela. O usuário, através do *menu*, pode selecionar os resultados e visualizá-los na tela do

dispositivo. A Figura 5.18 mostra o SisBDR informando o usuário que as consultas foram realizadas com sucesso e que os resultado podem ser visualizados a partir do *Menu*.

Figura 5.16 – A conexão com o servidor e as consultas estão sendo solicitadas

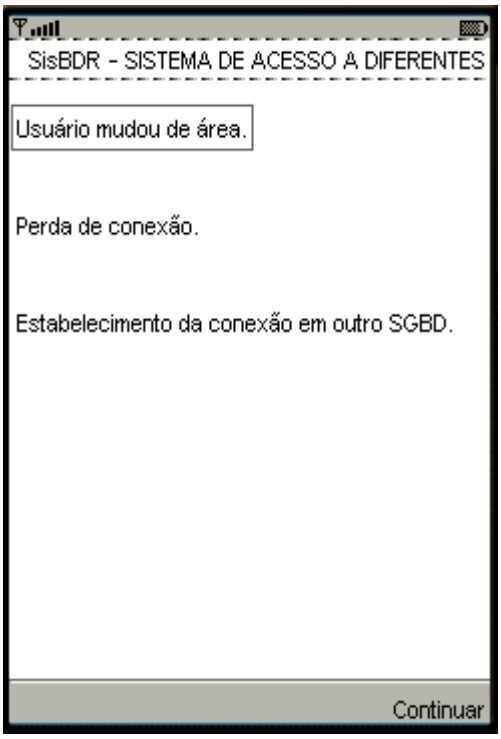

Figura 5.17 – Perda de conexão com o servidor

A Figura 5.19 e a Figura 5.20 mostram os resultados das consultas.

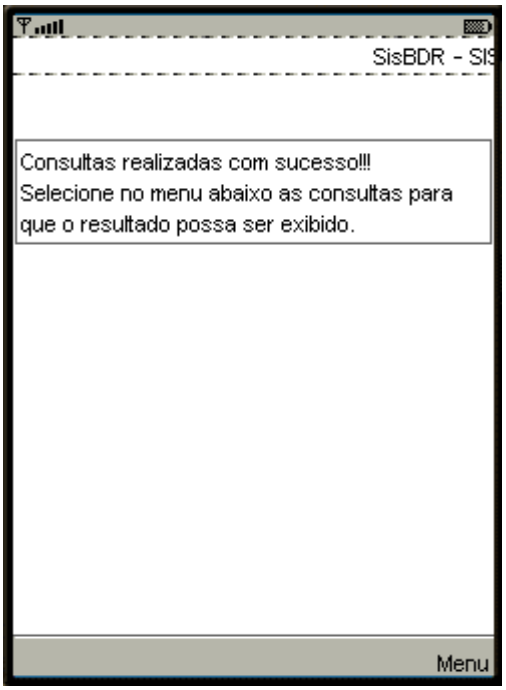

Figura 5.18 – Tela de informação ao usuário

Figura 5.19 – Resultado Consulta I

Figura 5.20 – Resultado Consulta II

5.10. Considerações Finais

O SisBDR pretende, além de uma contribuição no desenvolvimento de aplicações móveis, proporcionar uma resposta às consultas solicitadas pelos usuários, independente da perda de conexão de um servidor de banco de dados.

Com o uso da mobilidade, o usuário ao realizar uma consulta em um servidor de banco de dados, poderá estar se locomovendo enquanto a consulta é realizada. Problemas na conexão com os servidores poderão ocorrer devido à locomoção do usuário ou por algum problema com os servidores. Quando ocorre uma perda de conexão com um dos servidores o SisBDR se conecta em outro servidor e informa ao usuário que ocorreu uma perda de conexão com o servidor selecionado por ele e que foi estabelecida uma nova conexão com outro servidor e os resultados das consultas são retornados com êxito ao usuário.

Com o sistema pretende-se contribuir no desenvolvimento de aplicações móveis, no conforto dos usuários e no estudo e obtenção de novas tecnologias.

6. CONSIDERAÇÕES FINAIS

6.1. Conclusão

A utilização de dispositivos móveis tornou-se quase indispensável e, acima de tudo, acessível à maioria das pessoas. Atualmente, milhares de pessoas utilizam a tecnologia móvel, desde usuários com fins comerciais a crianças para jogos, mensagens, *Internet*, comunicação, dentre outros. A mobilidade se tornou parte do dia a dia das empresas e das pessoas, motivando esforços rumo a um maior investimento na área da tecnologia móvel.

A tecnologia móvel não é apenas uma invenção, ela pode ser considerada uma revolução, pois foi capaz de atingir o cotidiano das pessoas e fazer parte da vida delas, modificando suas rotinas e formas de tomar decisões. Apesar, de ser uma tecnologia recente, muitos projetos vêm sendo desenvolvidos com relação a aplicações móveis. A cada dia mais empresas estão surgindo no mercado da mobilidade. Percebe-se que no momento vem sendo um dos comércios mais bem sucedidos da economia mundial.

No desenvolvimento das aplicações móveis, o uso da tecnologia J2ME deixa os desenvolvedores livres para desenvolver aplicações e executá-las em qualquer dispositivo que possua uma máquina virtual, independente de sua capacidade computacional. Considerando essas diversas vantagens no desenvolvimento do SisBDR foi utilizada a tecnologia J2ME.

Além disso, os *WebServices* permitem a integração entre sistemas e compatibilidade de aplicações. Com esta tecnologia, é possível que novas aplicações possam interagir eficientemente com aquelas existentes e que sistemas desenvolvidos em plataformas diferentes sejam compatíveis. Com os *WebServices*, é possível a utilização de aplicações no modelo cliente/servidor e a possibilidade do cliente ser móvel, facilitando assim o desenvolvimento de aplicações para dispositivos móveis. Devido às limitações computacionais nos dispositivos, o processamento das operações são realizadas no servidor, que geralmente se localiza em um equipamento com alto poder de processamento, e os dispositivos móveis ficam encarregados de somente solicitar as operações e retornar os resultados das operações que foram realizadas nos servidores aos usuários.

6.2. Contribuição

A mobilidade, atualmente, é uma das tecnologias de maior investimentos nos últimos anos. A necessidade de disponibilizar informações cada vez mais rapidamente e onde quer que um usuário necessite faz com que o uso da rede sem fio cresça a cada dia e, com isso, surge a necessidade de aplicações suportarem essa tecnologia. Em vários dispositivos, a rede sem fio se faz presente, desde simples telefones ou dispositivos similares a computadores portáteis.

Com o SisBDR pretende-se contribuir no desenvolvimento de aplicações móveis e ao mesmo tempo estar contribuindo no aumento do uso da tecnologia móvel. O SisBDR permite ao usuário a realização de consultas em servidor de banco de dados de forma que a perda de conexão com um dos servidores é tratada de forma transparente ao usuário.

6.3. Trabalhos Futuros

Como trabalho futuro, podem-se desenvolver mais funções para o SisBDR, a ponto que ele, além das consultas, permita ao usuário realizar inserções, alterações e exclusões dos dados nos servidores através de um dispositivo móvel.

Aproveitando o caso de uso utilizado no desenvolvimento do SisBDR, uma funcionalidade de reserva a hospedagem pode ser implementada. Assim que o usuário selecione uma hospedagem o sistema faça uma conexão com a hospedagem e retorne a disponibilidade de vagas, permitindo ao usuário fazer reservas a partir de seu CPF, por exemplo.

Além disso, pode-se utilizar a idéia do SiSBDR para acessar dados armazenados em tecnologias de armazenamento diferentes (por exemplo, banco de dados relacional, banco de dados orientado a objetos e banco de dados objeto-relacional). Em conseqüência disso, permitir acessar dados em diferentes Sistemas Gerenciadores de Bancos de Dados diferentes de mesma tecnologia (por exemplo, *MySQL*, *SQL server*, *Firbird* e *Postgres*).

REFERÊNCIAS BIBLIOGRÁFICAS

[Almeida, 2004] Almeida, L. B. – **Introdução à J2ME e programação MIDP.** Mundo Java – Programando para Dispositivos Móveis. Número 5. Ano I.

[Amaral, 2002] Amaral, H. - **WEB DEVELOPMENT – *WebServices* com ColdFusion MX**. Developers' CIO Magazine. Ano 6 – N° 71, Julho/2002.

[Castaldelli, 2003] Castaldelli, M. – Teleco – Informações em Telecomunicações - **Aplicações Atuais e Futuras para *Internet* Móvel** Endereço: <http://www.teleco.com.br/tutoriais/tutorialcmovel/Default.asp>. Acessado em: Janeiro de 2006

[Caldeira, 2005] Caldeira, H. N. – **Java 2 Micro Edition – Do Pager ao PDA, com J2ME.** Java Magazine**.** Edição 1. Ano I, 2005.

[CET/UNB, 2007] Centro de Excelência em Turismo/ Universidade de Brasília – **PAC: investimentos de R$58,3 bilhões em logística beneficiam Turismo**. Endereço: <http://www.cet.unb.br/index.php/cet/turismo_na_midia_indice/pac_investimentos_de_r_58_3_bilhoes_em_logistica_beneficiam_turismo>. Acessado em: Março de 2007.

[Compera, 2006] Compera – Mobilidade Brasil. Setembro de 2002 - número 1 - ano 1. Endereço: <http://www.compera.com.br/Newsletter/MBrasil0101.html>. Acessado em: Dezembro de 2006.

[Devmedia, 2006] Devmedia Group, **Desenvolvendo uma aplicação J2ME**. Endereço: <http://www.devmedia.com.br/visualizacomponente.aspx?comp=4025&site=5>. Acessado em: Dezembro de 2006

[Freire, 2002] Freire, H. – ***WebServices*: A Nova Arquitetura da *Internet***. Developers' CIO Magazine. Ano 7 – N° 73, Setembro/2002.

[Gil, 1991] Gil, A. C. **Como Elaborar Projetos de Pesquisa** – São Paulo: Atlas, 1991.

[Gomes, 2007] Gomes, A. **Tutoria J2ME – Visão Geral**. Endereço: <http://www.mundooo.com.br/php/modules.php?name=MOOArtigos&pa=showpage&pid=9>. Acessado em: Fevereiro de 2007.

[IDC, 2007] Instituto de Desenvolvimento Cultural. Endereço: <http://www.idc.pt/site>. Acessado em: Março de 2007.

[Júnior, 2002] Júnior, C. A. – **Construindo Aplicações Distribuídas com J2EE e .NET**. Developers' CIO Magazine. Ano 7 – N° 73, Setembro/2002.

[Lee, 2005] Lee, V.; Schneider, H.; Schell, R. – **Aplicações Móveis – Arquitetura, projeto e desenvolvimento.** 1ª. ed., São Paulo: Pearson Education do Brasil, 2005.

[Luz, 2005] Luz, M., Wildt, D., Trevisan, A. – **J2Me em Campo – Acompanhamento de serviços por celular.** Java Magazine. Edição 15. Ano II, 2005.

[Muchow, 2004] Muchow, J. W. – **Core J2ME - Technology & MIDP.** 1ª. ed., São Paulo: Pearson Education do Brasil, 2004.

[Pamplona, 2006] Pamplonca, V., F. – Javafree.org. *WebServices*. **Construindo, disponibilizando e acessando *WebServices* via J2SE e J2ME**. www.javafree.org. Novembro de 2006.

[Pereira, 2004] Pereira, F. M. Q. , Bigonha, R. S., Valente, M. T. O., Bigonha, M. A. S. **Chamada Remota de Métodos na Plataforma J2ME/CLDC –** Revista Cientifica Periódica – Telecomunicações. Endereço: <http://compilers.cs.ucla.edu/fernando/publications/journals/Inatel2004.pdf>. Acessado em: Fevereiro de 2007.

[Rischpater, 2001] Rischpater, R. – **Desenvolvendo WIRELESS para WEB – Como Enfrentar os Desafios dos Projetos para a Web Sem Fio.** 1ª. ed., São Paulo: MAKRON *Books*, 2001.

[Sun, 2006] Sun Microsystems, Inc. **Mobile Information Device Profile**. Endereço: <java.sun.com/products/midp/midp-ds.pdf>. Acessado em: Outubro de 2006.

[Sun, 2006 a] Sun Microsystems, Inc.. **Java 2 Plataform, Micro Edition, Data Sheet**. Endereço: <http://java.sun.com/j2me/j2me-ds.pdf>. Acessado em: Outubro de 2006.

[Topley, 2002] Topley, K. – **J2ME in a Nutshell.**. ed., 2002.

[Thiollent, 1994] THIOLLENT, M. **Metodologia da Pesquisa-Ação**, São Paulo: Cortez, Sexta Edição, 1994.

[Urresti, 2002] Urresti, H. R. – **Tecnologia de *WebServices*: Passado, Presente e Futuro**. Developers' CIO Magazine. Ano 7 – N° 73, Setembro/2002.

Anexo A – Instalando o SisBDR

A.1 Aplicação Servidor do SisBDR

Na aplicação servidor do SisBDR, é necessário a instalação de um servidor *Web*. Para isso, será utilizado *Tomcat*. *Tomcat* é um servidor de aplicações Java para *Web* e é *software* livre criado pela *Sun* para as tecnologias *Java Servlet* e *JavaServer Pages* (JSP). *Tomcat* é inteiramente escrito em Java e, portanto, necessita de uma *Java Virtual Machine* (JVM) — Máquina Virtual Java — para ser executado. Assim, é necessário ter a plataforma *Java 2 Platform Standard Edition* (J2SE) instalada. A seguir, é apresentado o procedimento de instalação do servidor *Tomcat*:

- **Passo 1**: Instalação da Máquina Virtual.

  Primeiramente, é necessário ter a JVM. Caso a JVM não esteja instalada, é possível fazer o *download* através do *site* da Sun (http://java.sun.com/downloads/index.html), lembrando que é a plataforma Java é livre, sendo grátis o seu *download*.

  Complementando a instalação do Java 2 SDK, deve-se definir a variável de ambiente JAVA_HOME apontando para seu local de instalação. Esta variável de ambiente padrão é usada pelo *Tomcat* e por vários outros sistemas baseados em Java, para determinar a JVM preferencial. Isto é muito importante se houver mais de uma instalação de J2SE no computador, mas a variável JAVA_HOME deve ser definida mesmo se houver apenas uma versão instalada. Para a configuração, deve-se acessar Painel de Controle → Sistema → aba Avançado. Deve-se pressionar o botão Variáveis de ambiente. No grupo, Variáveis do sistema (recomendado, pois fica valendo para todos os usuários), deve-se criar uma nova variável JAVA_HOME ou editá-la, caso já exista, definindo como valor o caminho da pasta de instalação conforme o caminho de instalação, por exemplo, C:\Arquivos de programas\Java\jdk1.5.0_9. A nova configuração passa a valer após o OK, exceto para janelas de DOS abertas.

- **Passo 2**: Instalando o servidor *Tomcat*.

  1. fazer o *download* do servidor em http://tomcat.apache.org/download-55.cgi;
  2. iniciar o programa instalador. O assistente de instalação será iniciado. No diálogo de Boas-vindas, clicar *Next*;
  3. concordando com os termos da licença do produto de *software*, clicar *I Agree* para prosseguir. O *Tomcat* é *software* aberto, de re-distribuição e uso livre e gratuito;

4. na escolha de componentes do *software*, o padrão é o tipo de instalação Normal, que inclui, além do núcleo essencial (core) do *Tomcat Servlet container*, a documentação, exemplos de *Servlet* e JSP e os ícones no *Menu* Iniciar. Esta seleção é adequada para o uso geral;
5. a seguir, será confirmado o local de instalação do *software*. Confirmar a pasta principal onde o *Tomcat* será instalado e clicar *Next* para prosseguir;
6. o diálogo de Configuração permite definir duas opções administrativas do *Tomcat*: o porto de rede pelo qual o *Tomcat* atenderá as requisições HTTP, funcionando como um servidor *Web* com o propósito de testes e administração do *Tomcat*, e o usuário e senha para o acesso à Administração do *Tomcat*;
7. o instalador do *Tomcat* procura detectar uma versão de *Java Runtime* (JRE) instalada, necessária para seu funcionamento, conforme a Figura A.1;

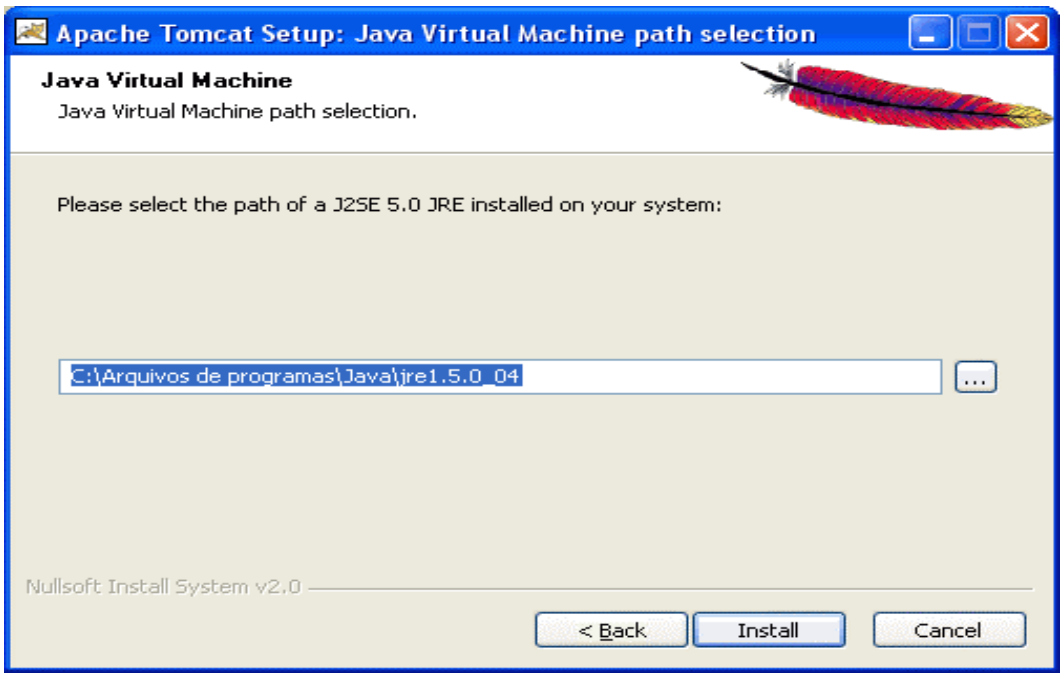

Figura A.1 – O instalador do *Tomcat* procura detectar uma versão de JRE

8. completadas as informações solicitadas pelo assistente, clicar em "*Install*" e aguardar o término da instalação. Ao término da instalação, o assistente apresentará as opções de executar o *Tomcat* pela primeira vez e de visualizar o Leia-Me. A Figura A.2 ilustra a operação do assistente;
9. iniciar o *Apache Service Manager*, utilizando o atalho em: Iniciar → Programas → Apache Tomcat → Monitor Tomcat. Deve surgir um pequeno ícone na área de notificação da barra de tarefas do *Windows* (ao lado do relógio). Este ícone indica o

estado atual do serviço *Tomcat* (quadrado vermelho = parado; triângulo verde = iniciado);

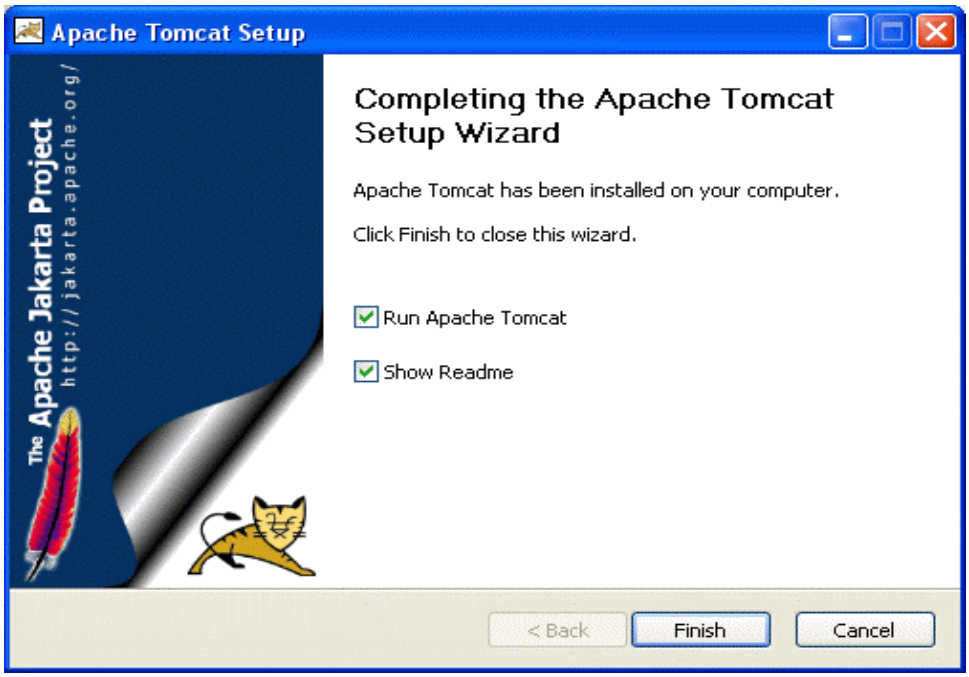

Figura A.2 – O assistente permite executar o Tomcat pela primeira vez

10. clicar no ícone com o botão direito do *mouse*; no *menu* de contexto que se abre, escolher *Start service* ou *Stop service*;
11. para testar se o *Tomcat* está rodando corretamente depois de iniciado, abrir o *browser* e ir para o endereço: http://localhost:8080/.

Para a execução do SisBDR, é necessário que os servidores de banco de dados contenham dados armazenados. Para a manutenção dos dados nos servidores, foi utilizado um Sistema de Cadastro de Hospedagens e o usuário, ao utilizar SisBDR através de um dispositivo móvel, poderá fazer consultas as hospedagens.

Nos servidores de banco de dados, foi utilizado o SGBD MySQL que possui como benefícios visíveis o fato de ser livre e manter a robustez necessária para o gerenciamento das informações.

A instalação do MySQL é simples. Deve-se fazer o *download* do SGBD em http://dev.mysql.com/downloads. Após a instalação, criar duas bases de dados, SP e MG, ambas com os campos nome, telefone, cidade, estado e tipo da hospedagem. Em seguida, deve-se utilizar o Sistema de Cadastro de Hospedagens.

Com o *Tomcat* instalado e os servidores de banco de dados contendo informações sobre as hospedagens, os seguintes passos são necessário para a instalação do SisBDR:

1. abrir o *browser* e digitar o endereço: http://localhost:8080/, o *Tomcat* será executado;
2. abrir a pasta que contém os arquivos da aplicação servidor do SisBDR e copiar o arquivo *ServiceSisBDR.war* que se encontra em *ServiceSisBDR/dis* para a pasta *manager*. Esta pasta está em *Apache Software Foundation/Tomcat /webapps*;
3. clicar *Tomcat Manager* que se encontra do lado esquerdo do navegador (Figura A.3);
4. no campo *WAR file to deploy*, buscar o arquivo que foi copiado para o *Tomcat* e clicar em *Deploy*. Com isso, a aplicação será instalada e executada no servidor (Figura A.4).

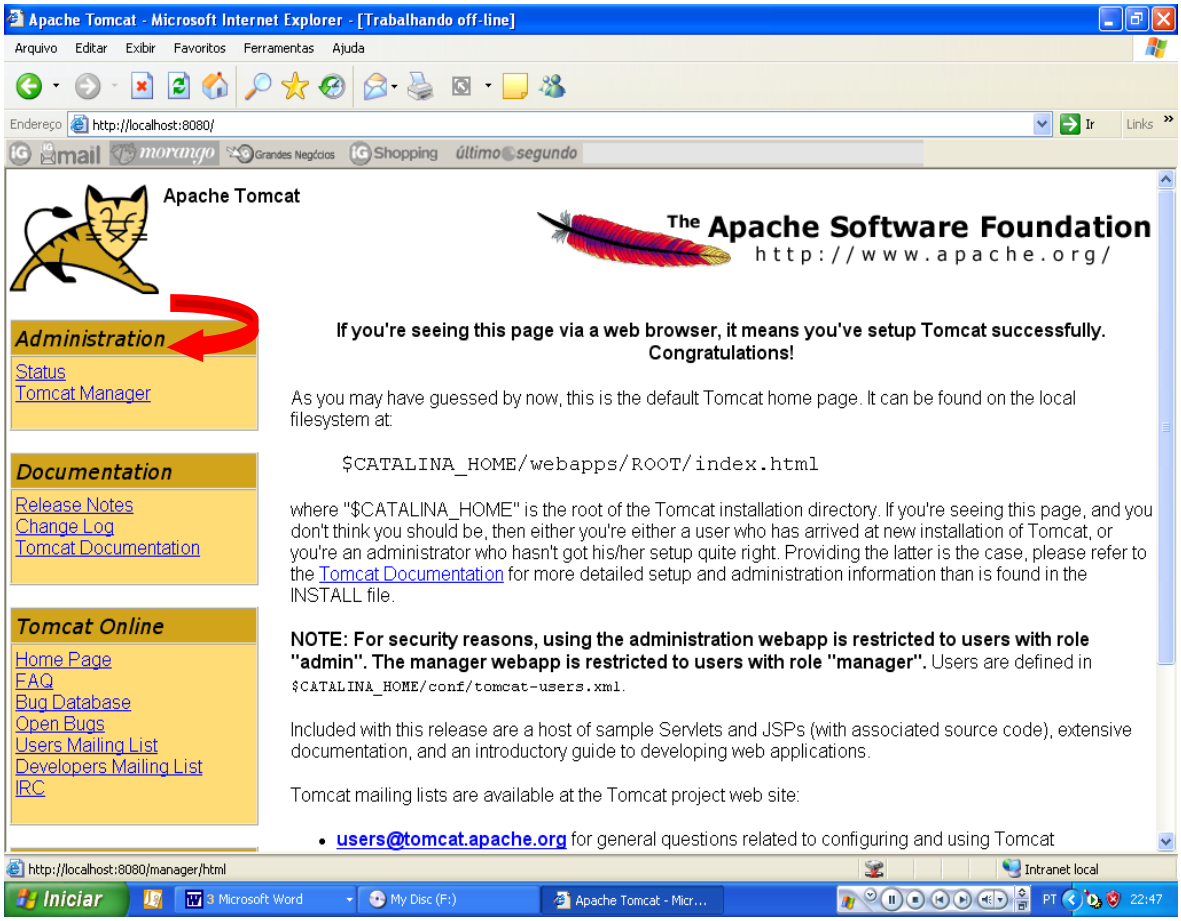

Figura A.3 – Opção *Tomcat Manager* do servidor Tomcat

A.2 Aplicação Cliente do SisBDR

Existem várias ferramentas que nos permitem desenvolver aplicações para o MIDP. A Sun tem, por exemplo, o *Sun ONE Studio, Mobile Edition*, mas existem muitos outros como o *CodeWarrior* da *Metrowerks*, o *WebSphere Studio Device Developer* da *IBM*, dentre outras. No SisBDR foi utilizada a ferramenta da Sun: o *Java ME Wireless Toolkit* –

WTK, sendo uma ferramenta muito simples no mundo da programação para dispositivos móveis.

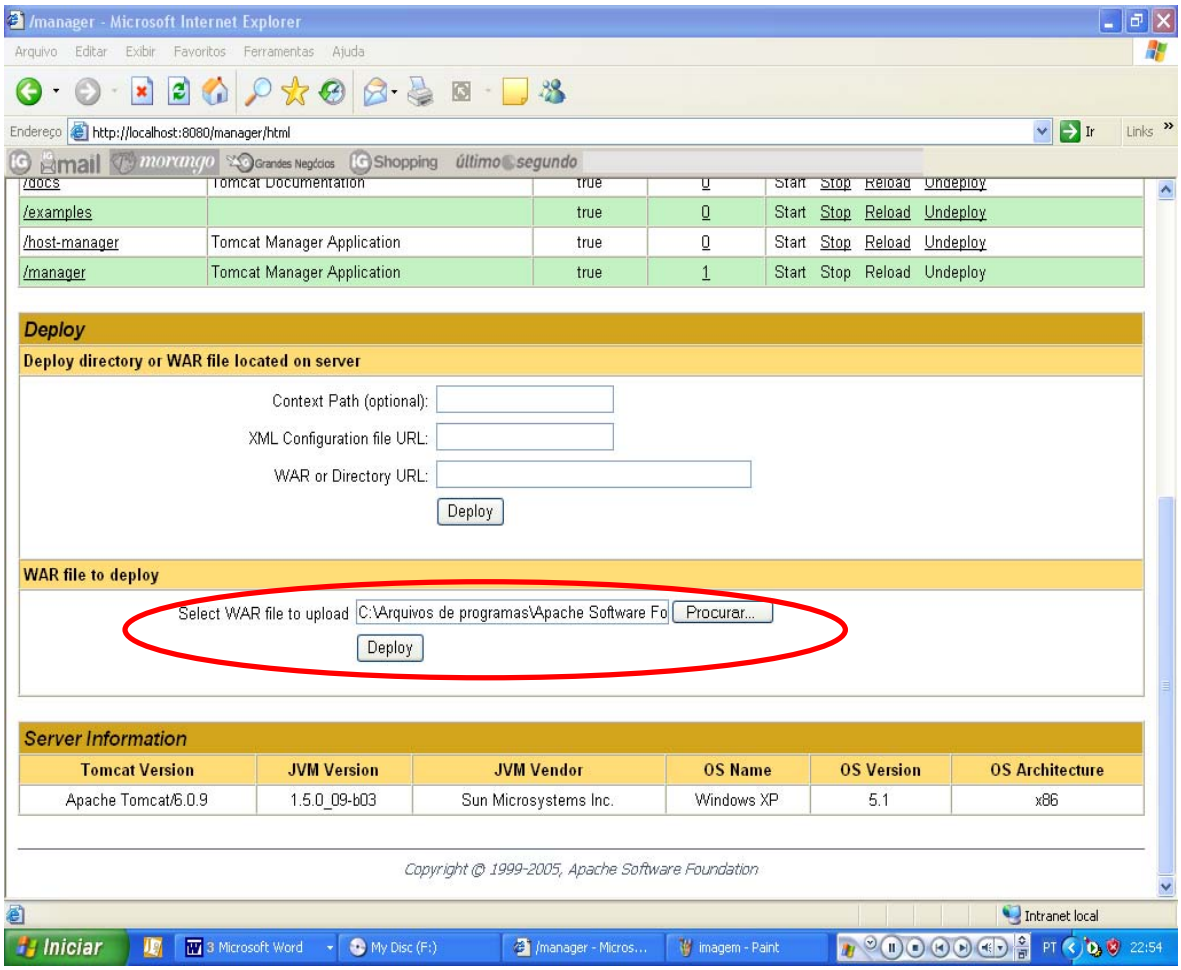

Figura A.4 – Fazendo *Deploy* no SisBDR

O WTK é uma aplicação gráfica que permite criar, compilar e executar projetos MIDP. Esta ferramenta não possui, no entanto, nenhum editor de texto pelo que a escrita do código tem de ser feita em outro programa. O WTK é uma ferramenta grátis e pode ser baixado a partir do site http://java.sun.com/products/j2mewtoolkit/.

A instalação do *Wireless Toolkit* é bem simples, o único cuidado é selecionar o diretório correto de sua JDK. A seguir, segue o procedimento de instalação do WTK:

1. fazer o *download* do WTK no *site* http://java.sun.com/products/j2mewtoolkit/;
2. iniciar o programa instalador. O assistente de instalação será iniciado. No diálogo de boas-vindas, clicar *Next*;
3. concordando com os termos da licença, clicar *Yes* para prosseguir. O WTK é um produto de *software* aberto, de re-distribuição e uso (comercial ou não) livre e gratuito;

4. o instalador do WTK detecta uma versão igual a 1.4.2 ou mais atualizada de Java 2 SDK instalada, necessária para seu funcionamento. Como J2SDK foi instalada antes do servidor *Tomcat*, pressupõe-se que será encontrado. Em seguida, clicar em *Next*. Caso a versão seja inferior a solicitada pelo WTK ou não seja encontrada, não será permitida a continuação da instalação do WTK;
5. a confirmação do local de instalação do produto de *software* será solicitada. Confirmar a pasta principal onde o WTK será instalado e clicar *Next* para prosseguir;
6. na próxima tela, é apresentada ao usuário uma sugestão do nome do produto de *software* que será afixado em Iniciar → Programas e, em seguida, clicar *Next*. Depois, é apresentado o local em que será instalado o WTK e local onde o produto de *software* localizou a JVM, concordando clicar *Next*;
7. completadas as informações solicitadas pelo assistente, clicar *Finish* e aguardar o término da instalação.

Com o WTK instalado, o usuário deve abrir a pasta que contém os arquivos da aplicação móvel do SisBDR e executar o arquivo *Mobile-SisBDR* localizado na pasta *dist*. Assim que o usuário executar o SisBDR é apresentado uma tela que permite trocar a URL do servidor caso necessário. A Figura A.5 ilustra a URL modificada no SisBDR.

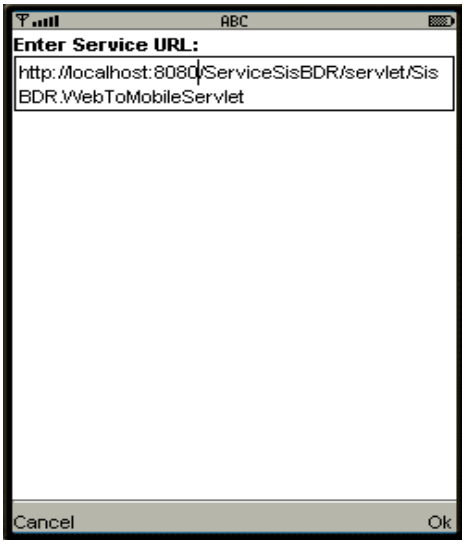

Figura A.5 – Modificação da porta do servidor

Anexo B – Código Fonte do SisBDR

Este anexo apresenta partes do código-fonte do Sistema de Acesso a Diferentes Bases de Dados Remotas Utilizando Tecnologia Móvel – SisBDR.

A Figura B.6 ilustra parte do código da classe *WebToMobileClientMIDlet*, localizada na aplicação cliente do SisBDR. Esta classe é responsável pela interface gráfica da aplicação, pela solicitação da conexão com o *WebService* e pela passagem dos parâmetros para as consultas a serem realizadas nos servidores de banco de dados.

```
    private WebToMobileClient client = new WebToMobileClient();

    private void setServiceURL(String serviceURL) {
        client = new WebToMobileClient(serviceURL);
    }

    private java.lang.String[] Consultar_returnValue; // field for holding value returned from method
    private java.lang.String[] Consultar_ConsultaII_returnValue; // field for holding value returned from method

// method calling the generated method
    private void call_Consultar() throws Exception {

       java.lang.String bd = "";
       java.lang.String nome = "";
       java.lang.String tipo="";

        if (choiceGroup1.getSelectedIndex()==0){
            bd = "SP";
        }else if (choiceGroup1.getSelectedIndex()==1){
           bd = "MG";
        }
        if (choiceGroup2.getSelectedIndex()==0){
            nome = "SP";
        }else if (choiceGroup2.getSelectedIndex()==1){
            nome = "MG";
        }
        if (choiceGroup3.getSelectedIndex()==0){
            tipo = "Hotel";
        }else if (choiceGroup3.getSelectedIndex()==1){
            tipo = "Pousada";
        }

       Consultar_returnValue = client.Consultar(nome, bd, tipo);
       Consultar_ConsultaII_returnValue = client.Consultar(nome, bd, tipo);
    }
```

Figura B.6 – Classe *WebToMobileClientMIDlet*

A Figura B.7 apresenta a classe *WebServiceSisBDR* que possui as operações a serem realizadas pelo *WebService*; no caso do SisBDR, somente a operação consulta é considerada. Após estabelecida a conexão, o *WebService* passa como parâmetro as consultas a serem realizadas para a classe *SisBDR* (Figura B.8), a qual é responsável pela solicitação da conexão com os servidores de banco de dados (Figura B.9). Estas classes pertencem a aplicação servidor do SisBDR.

```
/*
 * WebServiceSisBDR.java
 *
 * Created on 13 de Dezembro de 2006, 21:02
 *
 * To change this template, choose Tools | Template Manager
 * and open the template in the editor.
 */

package SisBDR;

import java.rmi.RemoteException;
import javax.jws.WebService;
import javax.jws.WebMethod;
import javax.jws.WebParam;

/**
 *
 * @author patricia
 */
@WebService()
public class WebServiceSisBDR {

    /**
     * Web service operation
     */
    public String[] Consultar(String nome, String bd, String tipo) throws RemoteException {
        // TODO implement operation
        return SisBDR.consultar(nome, bd, tipo);
    }



}
```

Figura B.7 – Classe *WebServiceSisBDR*

```
sql = "SELECT * FROM hospedagem WHERE estado LIKE '" +nome+ "%'";
sql2 = "SELECT * FROM hospedagem WHERE tipo LIKE '" +tipoHospedagem+ "%'";
try {
    if (bd.equals("SP")){
        conexao.setDataBase("SP");

    }else {
        conexao.setDataBase("MG");

    }
    conexao.conectar();
    resultado = conexao.consultaSQL(sql);
    int aux = 0;     // variavel auxiliar

    dados[aux] = "              Consulta pelo estado: ";
    while (resultado.next())  {
        aux = aux + 1;
        dados[aux] = "Nome da Hospedagem: " + resultado.getString("nome");
        aux = aux + 1;
        dados[aux] = "Telefone: " + resultado.getString("telefone");
        aux = aux + 1;
       .
       .
       .
    }
    conexao.desconectar();
    if (bd.equals("SP")){
        conexao.setDataBase("MG");

    }else {
        conexao.setDataBase("SP");

    }
    conexao.conectar();
    resultado2 = conexao.consultaSQL(sql2);

    //A variavel eh inicilizada com 25 pois o Sistema pretende exibir somente alguns resultados obtidos
    //nas consultas devido a limitação do tamanho da tela dos dispositivos.
    aux=25;

    dados[aux+1] = "                  Consulta pelo tipo";
    aux=aux +1 ;
    while (resultado2.next())  {
        aux = aux + 1;
        dados[aux] = "Nome da Hospedagem: " + resultado2.getString("nome");
        aux = aux + 1;
        .
        .
        .
    }

    for (int a = aux+1; a<51; a ++){
        dados[a]=" ";
    }
   conexao.desconectar();
} catch (SQLException ex) {
    ex.printStackTrace();
}

return dados;
```

Figura B.8 – Classe SisBDR

```
//Construtores
public Conexao() throws ClassNotFoundException {
    Class.forName("com.mysql.jdbc.Driver");
}

public Conexao(String host, String database) throws ClassNotFoundExceptior
    setHost(host);
    setDataBase(database);
    Class.forName("com.mysql.jdbc.Driver");
}

//Métodos de acesso
public Connection getConexao() {
    return conexao;
}

public Statement getStmt() {
    return stmt;
}

//Outros métodos
public void conectar() throws SQLException {
    String url = "jdbc:mysql://" + getHost() + "/" + getDataBase();
    this.conexao = DriverManager.getConnection(url, user, password);
    this.stmt = getConexao().createStatement();
}

public void conectar(String usuario, String senha) throws SQLException {
    String url = "jdbc:mysql://" + getHost() + "/" + getDataBase();
    this.conexao = DriverManager.getConnection(url, usuario, senha);
    this.stmt = getConexao().createStatement();
}

public void desconectar() throws SQLException {
    getStmt().close();
    getConexao().close();
}

public ResultSet consultaSQL(String sql) throws SQLException {
    ResultSet resultado;
    if (getStmt().execute(sql)) {
        resultado = getStmt().getResultSet();
    } else {
        return null;
    }
    return resultado;
}

public int ExecutarSQL(String sql) throws SQLException {
        return getStmt().executeUpdate(sql);
}
```

Figura B.9 – Classe C*onexao*