Acesso a Bancos de Dados através de Celulares

Daniel Rosa Soares[1], Diego Kreutz[1], Gustavo Zanini Kantorski[1]

[1]Curso de Sistemas de Informação – Universidade Luterana do Brasil (ULBRA)
Santa Maria – RS – Brasil

{diego.ulbrasm, danielgrs}@gmail.com, gustavoz@cpd.ufsm.br

Abstract. *The approach each bigger time of the people in the access information, not only of passive form, but also of active form, generating knowledge is basic. The development of tools with mobility helps the access to the knowledge for the people. This article shows an implementation for the application development with mobility, specifically cellular, for database access.*

Resumo: *A aproximação cada vez maior das pessoas no acesso a informações, não somente de forma passiva, mas também de forma ativa, gerando conhecimento é fundamental. O desenvolvimento de ferramentas com mobilidade ajuda o acesso ao conhecimento pelas pessoas. Este artigo apresenta uma implementação para acesso a banco de dados com mobilidade, especificamente aparelhos celulares.*

1. Introdução

A rápida popularização dos dispositivos móveis (celulares, palm-tops, pagers, etc.) vem estimulando a integração de serviços disponibilizados na Internet em pequenos dispositivos. Um estudo recente revela que o número de usuários de dispositivos móveis tem crescido a taxas entre 30% e 50% por ano, a maioria em redes de celulares. Este crescimento se dá devido às facilidades que a tecnologia de comunicação sem fio e os dispositivos móveis proporcionam no dia-a-dia, tais como, navegação em sites da *Web*, visualização e edição de documentos, envio e recebimento de mensagens, acesso a sistemas de informação em geral [Villela 2006].

Juntamente com a popularização dos celulares, a população vem participando ativamente no acesso ao conhecimento. Informações armazenadas em grandes bancos de dados passam a ser compartilhadas através de redes de computadores e aplicações em computadores. O desenvolvimento de interfaces que permitam o acesso a bancos de dados através de dispositivos móveis e computação ubíqua passa a ser de grande importância no momento que essa participação passa a ser de maneira universal.

Este artigo apresenta uma implementação para acesso a bancos de dados através de celular. A implementação foi desenvolvida utilizando tecnologias como Java, J2ME, PHP, WAP e bancos de dados. O acesso foi desenvolvido utilizando uma aplicação J2ME e uma aplicação WAP.

A próxima seção apresenta as tecnologias utilizadas para a implementação do trabalho. Na seção 3 são apresentados o acesso ao banco de dados e a implementação dos aplicativos. Considerações finais entre aplicativos J2ME e WAP são apresentadas na seção 4.

2. Tecnologias Utilizadas

Nesta seção, serão abordadas as tecnologias utilizadas para o desenvolvimento do trabalho, como linguagens e ambientes utilizados, apresentando suas características e propriedades.

2.1. Java Micro Edition

A linguagem Java tem como premissa o fato de ser executada em diferentes dispositivos, mas em 1999, passou a destinar uma atenção especial à plataforma J2ME, para terminais com limitação de memória e processamento. O J2ME segue o princípio da linguagem Java, "*Write Once, Run Anywhere*", permitindo a programação para celulares independente de modelo, tipo ou fabricante do aparelho. Isso significa que a portabilidade do código, prevista desde a origem da linguagem, continua sendo seguida nas plataformas para dispositivos de pequeno porte.

Cada máquina virtual é desenvolvida pelo fabricante do aparelho, portanto, o aplicativo J2ME se preocupa apenas em interagir com a máquina e não com o dispositivo real. A máquina virtual para dispositivos de pequeno porte é diferente em relação a máquina padrão, devido a questões de limitação de processamento dos aparelhos celulares. A máquina virtual para dispositivos pequenos é denominada *Kilobyte Virtual Machine* (KVM) e possui o tamanho de aproximadamente 8Kbytes.

Para resolver o problema da diversidade de dispositivos eletrônicos, foram definidos dois conceitos: as *configurações* e os *perfis*.

Nas configurações, são definidos o vídeo, a memória e a conectividade de rede (ou limitações disso) disponível no dispositivo [Muchow 2004]. Existem duas configurações disponíveis: CDC (Configuração de Dispositivo Conectado) e CLDC (Configuração de Dispositivo Conectado Limitado). Na tabela 1, pode ser visto um resumo comparativo das duas configurações, observando-se que a configuração CDC consome mais memória, tanto para executar o Java, quanto para alocar memória em tempo de execução, além de precisar uma conectividade de rede persistente e alta em comparação com a configuração CLDC.

Tabela 1. Características dos Dispositivos

Características	**Configuração de Dispositivo Conectado (CDC)**	**Configuração de Dispositivo Conectado Limitado (CLDC)**
Memória	512 Kb	128 Kb
Alocação de Memória	256 Kb	32 Kb
Conectividade	Persistente e Alta	Intermitente e Baixa

Além das configurações, o conceito de perfil foi criado devido às variações de recursos entre cada celular, fornecendo bibliotecas para que sejam desenvolvidos aplicativos para um modelo particular de dispositivo. Os tipos de perfis são definidos através do *Mobile Information Device Profile* (MIDP).

2.2. PHP

O PHP é uma linguagem de código-fonte aberta, muito utilizada na Internet e, especialmente criada para o desenvolvimento de aplicativos *Web*. *Hypertext Preprocessor* atualmente é a sigla de PHP e pode ser considerada uma combinação de linguagem de programação e servidor de aplicações [PHP 2006].

A linguagem PHP foi escolhida para desenvolver o *Web Service*, pois além de ser uma linguagem *open source*, atingiu as expectativas no desenvolvimento. Além disso, o PHP pode ser utilizado na maioria dos sistemas operacionais, como: Linux, variantes Unix, Microsoft Windows, Mac OS, RISC OS e possui suporte muito amplo para banco de dados, como *dBase*, *Interbase*, *mSQL*, *mySQL*, *Oracle*, PostgreSQL, entre outros [Rocha, 2003].

Um *Web Service* é um sistema de software identificado por um URI, cujas interfaces públicas e ligações são definidas e descritas usando XML (*eXtensible Markup Language*). Sua definição pode ser descoberta através de outros sistemas de *software*. Estes sistemas podem então interagir com o serviço Web de uma maneira prescrita pela sua definição, usando mensagens baseadas em XML cujo transporte é feito por meio dos protocolos da Internet [SINGH 2006]. Assim, é possível chegar a uma definição mais simples, como: um *Web Service* é um aplicativo de *software*, acessível na Intranet através de uma URL [WebService 2006].

2.3. WML (*Wireless Markup Language*)

WML é uma linguagem de marcação baseada em XML própria, onde pode ser exibida em *mini-browsers* utilizando a tecnologia WAP (*Wireless Application Protocol*). A tecnologia WAP fornece um método-padrão de acesso a conteúdos na Internet, a partir de dispositivos sem fio [Muchow, 2004].

A título de analogia pode-se dizer que WML é o HTML dos portáteis sem fios. A maioria das *tags* WML é igual as do HTML. No entanto, sendo WML uma aplicação XML é uma linguagem bem mais rigorosa que HTML [Samy 2004].

2.4 Banco de Dados

Foram selecionados dois bancos de dados para o desenvolvimento dos aplicativos. Os bancos escolhidos foram o PostgreSQL [Postgresql 2006] e o MySQL [Mysql 2006] por serem *open source* e *free*. No entanto, qualquer banco de dados que suporte a linguagem ANSI/SQL pode ser utilizado.

3. Acesso a Bancos de Dados através de Celulares

O aplicativo desenvolvido possibilita que um cliente, utilizando um dispositivo móvel, através de um *Web Service*, possa acessar uma base de dados utilizando um aplicativo J2ME, instalado no próprio aparelho, conforme apresentado na figura 1a, ou através de um *minibrowser*, utilizando WAP, conforme mostrado na figura 1b.

O estudo de caso foi baseado em um sistema que tem por objetivo prover informações sobre a situação de veículos através de um celular. Um exemplo prático de aplicação seriam os guardas de trânsito e policiais rodoviários, que através de uma consulta simples, por intermédio de um celular, podem verificar a situação de um determinado

veículo. Informações podem incluir, por exemplo, dados que indicam se o veículo é roubado ou não, e se o emplacamento está em dia.

Figura 1. Modelo de Implementação.

Para operacionalizar o sistema, é necessário um banco de dados com o cadastro dos proprietários e seus respectivos veículos, possibilitando assim realizar consulta sobre a situação atual de cada automóvel cadastrado. O resultado da consulta é então exibido no *display* do celular.

3.1 Aplicativo J2ME

Um aplicativo J2ME possui como característica a portabilidade, pois utiliza máquinas virtuais java, como a KVM. A figura 2 ilustra uma aplicação sendo executada no celular. Como pode ser visto a aplicação é executada pela KVM que por sua vez está em execução no sistema operacional nativo do celular.

Toda aplicação J2ME possui um conjunto de funcionalidades pré-definidas pelo MIDlet (aplicação desenvolvida para celular) e um ciclo de vida. A primeira fase de vida é o estado *startApp* (Ativa, ou seja, momento em que a aplicação é inicializada), podendo passar para dois estados distintos, *destroyApp* (Destruída), finalizando a aplicação, ou *pauseApp* (Pausada), podendo retornar ao estado ativo.

Figura 2. Camadas de um aplicação J2ME.

A fim de resolver o problema proposto utilizando J2ME, é necessário definir a arquitetura do aplicativo. Como o objetivo é estudar consultas a banco de dados através de celulares, descarta-se a hipótese de ter o banco de dados no celular, ou seja, a aplicação terá que ter, no mínimo, duas camadas, uma cliente que é executada no

dispositivo móvel, e outra servidora que faz a interface com o banco de dados. Considerando esse aspecto, a figura 3 apresenta as camadas definidas para a arquitetura.

3.1.1 Implementação Cliente

A camada cliente da aplicação é uma interface, que possibilita aos usuários realizar consultas a um banco de dados através do celular. Estas consultas são feitas através de um *Web Service*, que será responsável pela troca de dados entre cliente e servidor.

Para o estudo de caso, onde usuários (policiais rodoviários e/ou guardas municipais) possam ter uma maior flexibilidade para a consulta ao banco de dados, foram definidos três campos possíveis de consulta: nome do proprietário, placa ou chassi do veículo.

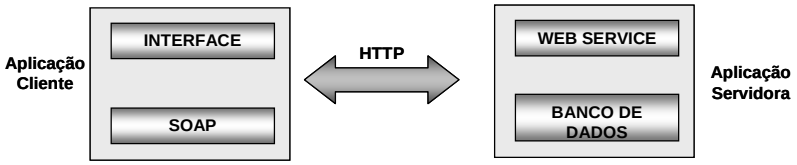

Figura 3. Arquitetura da Aplicação J2ME.

Escolhida a opção de consulta que o usuário deseja, é disponibilizada uma interface para entrada de dados. A partir da entrada de dados é criado um objeto SOAP (*Simple Object Access Protocol*) que irá fazer o transporte dos dados até a interface do servidor. A figura 4 ilustra todas as etapas desde a interface inicial, o acesso ao banco de dados e o retorno da resposta para o usuário.

Ao abrir o aplicativo, residente e instalado no aparelho celular, o usuário seleciona a opção (Nome, Placa ou Chassi) que será consultada, conforme apresentado na figura 4a. Os botões *OK* e *SAIR* possuem, respectivamente, as funcionalidades de inicialização do formulário de entrada de dados e acionamento do método que destrói a *MIDlet*.

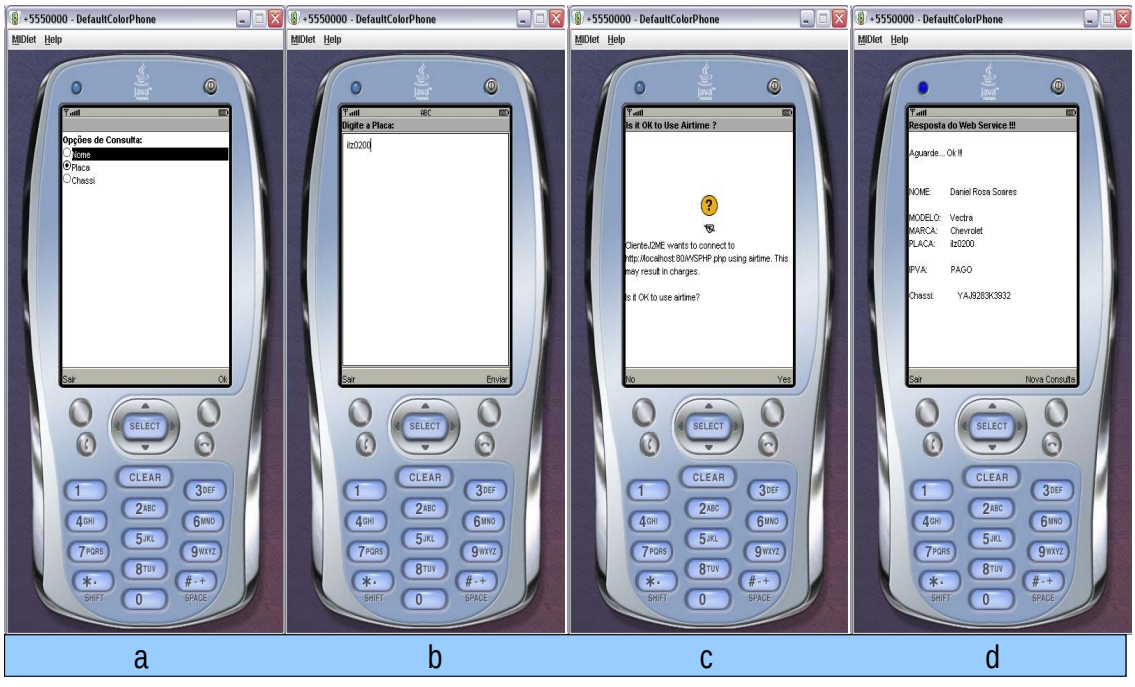

Figura 4. Etapas de Consulta ao Banco de Dados.

Após a escolha de um dos três itens (placa, nome ou chassi), e pressionado o botão *OK*, será apresentada ao usuário uma interface para a especificação do valor a ser procurado na base de dados (figura 4b). Na interface de entrada de dados são apresentados também os botões *ENVIAR* e *SAIR* (figura 4b), com as funcionalidades de enviar a informação digitada para o *Web Service* e de destruir a MIDlet, respectivamente. No momento que o botão *ENVIAR* (figura 4b) for ativado é criado um novo formulário, que será responsável pela exibição do estado de processamento da requisição do usuário. O resultado pode ser visualizado nas figuras 4c e 4d.

O código fonte, apresentado na figura 5, mostra onde são criados os componentes da figura 4a. Nas linhas 2, 3 e 4, são criados os componentes do menu e na linha 5 é definido qual desses componentes será o *default* ou padrão quando a aplicação for executada. Na linha 7 é adicionado ao formulário principal (formPrinc) o menu com as opções de consulta. Nas linhas 8 e 9 são adicionados os comandos (botões) Ok e Sair ao formulário principal. Na linha 11 o *display* é "setado" para que apareça o formulário principal, já com os componentes adicionados.

```
1   public void startApp() {
2     escolha.append("Nome", null);
3     escolhaPadrao = escolha.append("Placa", null);
4     escolha.append("Chassi", null);
5     escolha.setSelectedIndex(escolhaPadrao, true);
6      ...
7     choiceGroupIndex = formPrinc.append(escolha);
8     formPrinc.addCommand(okCommand);
9     formPrinc.addCommand(exitCommand);
10     ...
11    display.setCurrent(formPrinc);    }
```

Figura 5. Criação de Componentes.

A criação e transporte do objeto SOAP, que será entregue ao *Web Service* no lado servidor, são ilustrados na figura 6. Na linha 3 é criado um objeto SOAP, que recebe como parâmetro o identificador do *Web Service* (no caso a *url*, pois as URLs costumam ser, por natureza, identificadores únicos) e a função que será ativada no provedor do serviço. Na linha 4 é instanciado um objeto (*httransp*) que será responsável pelo transporte, através de http (*HyperText Transfer Protocol*), do objeto SOAP.

Os dados digitados pelo usuário são incluídos no objeto SOAP através do método *addProperty*, conforme ilustrado na linha 5 (figura 6). Este método recebe dois parâmetros, o nome da variável e o seu conteúdo. No caso, o nome da variável deve corresponder a variável esperada pela função do *Web Service* que será ativada e o valor representa o conteúdo informador pelo usuário, no caso, a variável *textodigitado*.

O próximo passo é o transporte do objeto SOAP até o provedor do serviço, o que é representado na linha 8. A chamada *httransp.call(cliente)* irá enviar os dados ao *Web Service* e aguardar uma resposta. O resultado será acrescentado a uma área de armazenamento temporário (*stBuff*) e posteriormente passado como parâmetro ao método *done* (linha 9), cuja função é exibir a resposta do *Web Service* no *display*. Caso ocorra algum problema com a chamada httransp.call(cliente), uma mensagem será gerada e igualmente exibida ao usuário (linha 11) através do método *done*.

```
1   public void run() {
2   ...
3     SoapObject cliente = new SoapObject(url, funcao);
4     HttpTransport httransp = new HttpTransport(url, funcao);
5     cliente.addProperty(tipo_cons, textodigitado);
```

```
6       ...
7           try {
8               stBuff.append(httransp.call(cliente));
9              done("\nAguarde... Ok !!!" + stBuff.toString());
10          } catch (Exception e) {
11           done("\n\n ERRO AO TENTAR EXECUTAR !");
```

Figura 6. Criação do objeto SOAP.

3.1.2 Implementação Servidor

Na interface cliente, após o usuário ter digitado os dados e solicitado a consulta ao banco de dados, as informações necessárias são encaminhadas até o provedor do serviço para processamento. O *Web Service*, no lado do servidor, é responsável por receber os dados do cliente, processá-los e enviar uma mensagem de resposta, conforme pode ser visto na figura 7.

Figura 7. Implementação do Web Service.

Quando a mensagem SOAP for recebida, os dados como função, variáveis e respectivos valores são extraídos e processados pelo serviço. São registradas as funções *consulta_placa* (Sserver->register('consulta_placa')) que receberá o dado a ser utilizado na pesquisa, ou seja, o identificador da placa desejada. Para a consulta aos tipos de dados nome e chassi existem outras duas funções igualmente registradas no *Web Service*, denominadas *consulta_nome* e *consulta_chassi*.

A figura 8 ilustra a conexão e consulta ao banco de dados. Nos experimentos realizados foi utilizado o PostreSQL, sendo este o motivo de o nome das funções iniciar pelos caracteres “pg”. Na linha 3 é feita a conexão com o banco, sendo passados à função *pg_connect*, o nome do banco (*dbname*), o nome do usuário (*user*) e a senha de acesso (*password*).

Nas linhas 4 e 5 é executada a consulta ao banco de dados. O valor enviado pelo cliente (contido na variável *$placa*) é utilizado na consulta, ou seja, o *select* é feito sobre o campo placa da tabela, onde o valor do campo é igual à entrada de dados do usuário no celular. A consulta, caso a placa for encontrada no banco, irá retornar os valores dos campos *nome*, *modelo*, *placa*, *ipva (imposto)* e *chassi*. Essas informações permitem, por exemplo, a um guarda de trânsito, verificar o estado de um veículo.

O resultado da consulta, armazenado na variável *$result*, será enviado como resposta ao pedido da aplicação cliente. O conteúdo da resposta é então exibido ao usuário, como ilustrado na figura 4d.

```
1  function consulta_placa($placa) {
2     ...
3     $con = pg_connect("dbname=ws1 user=postgres password=postgres");
4     $result = pg_exec("SELECT nome,modelo,marca,placa,ipva,chassi FROM
5  carros WHERE placa='$placa'");
```

Figura 8. Conexão e Consulta ao Banco de Dados.

3.2 Aplicativo WAP

Uma página WAP, diferentemente do aplicativo J2ME, não necessita de maiores componentes instalados no aparelho celular, mas continua tendo algumas restrições quanto ao dispositivo, como, utilização de alguns recursos que poderiam ser utilizados numa página HTML.

No caso do aplicativo WAP, não há implementação no lado do cliente. A única restrição consiste em o celular possuir um *minibrowser* capaz de interpretar documentos WML. Na figura 9 é apresentado o funcionamento do aplicativo WAP.

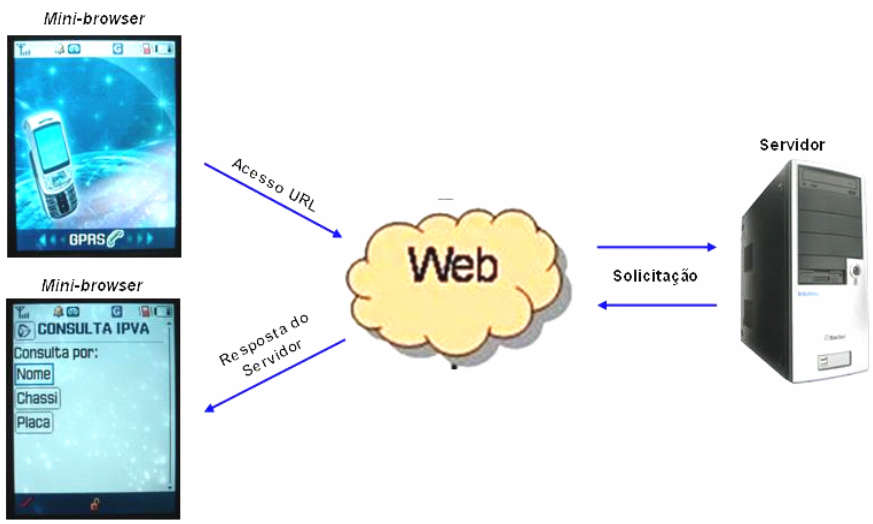

Figura 9. Acesso ao BD através de WAP.

Basicamente, esse modelo é mais simples. O usuário acessa, via *minibrowser*, o endereço onde está hospedada a página WML, como ilustrado na figura 10a. Em um segundo momento, selecionada uma opção de consulta, uma nova página WML será enviada ao *minibrowser*, possibilitando ao usuário entrar com os dados a serem pesquisados. A interface de entrada de dados pode ser vista na figura 10b. Por fim, os dados digitados pelo usuário são enviados ao servidor, para uma página PHP. Esta página contém o código que irá processar a consulta no banco de dados e exibir (retornar) o resultado ao usuário (figura 10c).

A identificação de uma página WML é realizada através de uma *tag*, chamada, *card*, sendo que várias páginas WML podem ser incluídas em um único arquivo, acessíveis como URL. O *id* (identificador) diferencia as várias *cards*, enquanto que o *title* (título) representa o cabeçalho da página que será exibida ao usuário. As ancoras (*anchor*), os

itens (tipos de dados, *input*) apresentados no *minibrowser*, na forma de menu, e as referências (*href*) aos diferentes *cards*, de acordo com o tipo de dados que poderá ser selecionado pelo usuário.

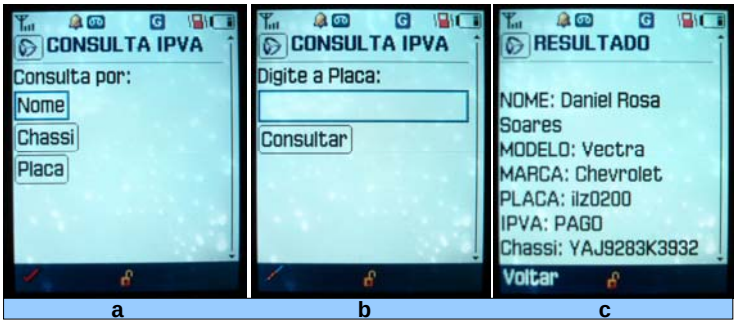

Figura 10. Interface do Aplicativo WAP.

Ao selecionar um dos itens (figura 10a) o usuário estará ativando uma *card* distinta. No entanto, basicamente, cada uma delas apresentará uma caixa de texto para entrada de dados. Na página WML está especificada o endereço da página PHP, que irá receber os dados digitados pelo usuário, o campo e o respectivo valor, os quais são utilizados para a consulta ao banco de dados. O código PHP responsável pela consulta ao banco de dados é apresentado na figura 11. As duas primeiras, e as duas últimas, linhas identificam o início e o final da página WML, que exibirá o resultado da consulta.

```
1   <wml>
2   <card id="resposta" title="RESULTADO">
3   <?php
4   …
5   $query = "SELECT nome,modelo,marca,placa,ipva,chassi
6                 FROM carros
7                 WHERE  placa='$placa'";
8   …….
9   $result = mysql_query($query);
10  if (mysql_num_rows($result) > 0){
11     while ($row = mysql_fetch_array($result)){
12        echo  "<br>NOME:  "  .  $row['nome']  .  "<br>MODELO:  "  .
13  $row['modelo'] "<br>MARCA: " . $row['marca'] . "<br>PLACA: " .
14  ……..
15  </card>
16  </wml>
```

Figura 11. Codigo PHP para acesso ao Banco de Dados.

4. Conclusões

A computação móvel aliada à área de banco de dados possibilita a aproximação da sociedade permitindo uma participação mais ativa da população e expandindo sistemas computacionais em locais onde, atualmente, não é possível a utilização de computadores.

Este artigo apresenta uma implementação para acesso a bancos de dados através de dispositivos móveis, mais especificamente, aparelhos celulares. Foram implementadas duas formas de acesso a banco de dados. A primeira através de um aplicativo que executa no próprio celular, utilizando a tecnologia J2ME, e a segunda através de um *minibrowser*, utilizando a tecnologia WAP.

Utilizando a tecnologia J2ME, o usuário deve possuir um aparelho celular que suporte a tecnologia Java, onde deve estar instalado uma KVM capaz de comportar o aplicativo. Desta forma, o usuário deve inicializar o aplicativo, através do menu do celular, para realizar o acesso ao banco de dados. Por outro lado, utilizando WAP, o usuário deve ter um celular com suporte a *minibrowser* e WML, acessando a página que fará a interface entre o usuário e o banco de dados.

Tanto a manutenção quanto a implementação do código foi mais fácil com o uso da tecnologia WAP. Ambas tecnologias possuem a dependência de sinal para utilização, visto que o banco de dados não está localizado no próprio aparelho.

O acesso ao aplicativo através de J2ME pode ser mais rápido, com um tempo de resposta melhor, visto que o mesmo está instalado no aparelho do usuário, enquanto na tecnologia WAP, um acesso através do *min-browser* poderá levar a várias requisições HTTP. Outra observação no acesso através de J2ME é o custo, pois a quantidade de tráfego na rede é menor, enquanto na tecnologia WAP mais informações devem ser transmitidas na rede.

Outra questão importante são os requisitos que os aparelhos devem possuir para executar o aplicativo. Para aplicações J2ME a quantidade de requisitos que os aparelhos devem possuir é maior que aplicações WAP.

Referências

Muchow, John W. *Core J2ME: Tecnologia & MIDP*. São Paulo: Pearson Makron Books, 2004.

Rocha, Cerli Antônio da. *Desenvolvendo web sites dinâmicos PHP, ASP e JSP*. Rio de Janeiro: Campus, 2003.

Singh, Inderjeet; Brydon, Sean; Murray, Greg; Ramachandran, Vijay; Violleau, Thierry; STEARNS, Beth. *Projetando Web Services com a Plataforma J2EE 1.4: Tecnologias JAX-RPC, SOAP e XML*. Rio de Janeiro: Ciência Moderna Ltda, 2006.

Villela, Marcos. *A briga esquenta*. Revista Connect. n. 6. ano 1. São Paulo: Motorpress Brasil, maio de 2006.

SAMY, Maurício. *Trabalhando com WML*. 2004. Disponível em: <http://www.crieseuwebsite.com/tutoriais/abretutorial.php?tutorial=93>. Acesso em: 10 dez. 2006.

POSTGRESQL. *Postgre SQL Documentation.* Disponível em : http://www.postgresql.org/docs/7.4/interactive/index.html. Acesso em: 18/12/2006.

MYSQL. *MySQL Documentation.* Disponível em: http://dev.mysql.com/doc/. Acesso em: 18/12/2006.

WEBSERVICE. *Web Services Architecture*. Disponível em: http://www.w3.org/TR/ws-arch/#gengag. Acesso em: 18/12/2006.

PHP. *PHP Documentation.* Disponível em : http://www.php.net/manual. Acesso em: 20/12/2006.