UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

PROTÓTIPO DE UMA APLICAÇÃO MÓVEL PARA LOCAÇÃO DE VEÍCULOS UTILIZANDO J2ME

MARCIANE SCHOTTEN

BLUMENAU
2011

2011/1-16

MARCIANE SCHOTTEN

PROTÓTIPO DE UMA APLICAÇÃO MÓVEL PARA LOCAÇÃO DE VEÍCULOS UTILIZANDO J2ME

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Sistemas de Informação - Bacharelado.

Prof. Ricardo Alencar de Azambuja, Mestre – Orientador

BLUMENAU
2011

2011/1-16

PROTÓTIPO DE UMA APLICAÇÃO MÓVEL PARA LOCAÇÃO DE VEÍCULOS UTILIZANDO J2ME

Por

MARCIANE SCHOTTEN

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Ricardo Alencar de Azambuja, Mestre – Orientador, FURB
Membro:	Prof. Francisco Adell Péricas, Mestre – FURB
Membro:	Prof. Roberto Heinzle, Doutor – FURB

Blumenau, 30 de julho de 2011.

Dedico este trabalho em memória do meu querido pai Rainoldo Schotten que mesmo não estando junto me impulsionou para a realização deste, a minha adorável mãe Lídia Feliciano Schotten pelo apoio incondicional em todos os momentos.

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça.

Tenho muito a agradecer a todos aqueles com quem convivi até hoje e que me trouxeram até aqui. Agradeço especialmente aos professores da FURB, amigos, com quem dividi as experiências e as idéias.

À minha mãe, que mesmo longe, sempre esteve presente. Aos meus irmãos Marisandra, Juliana, Carlos Alexandre e minha cunhada Eliane Luiz, que me apoiaram de diversas maneiras durante esta importante etapa da minha vida.

Em especial ao meu namorado, sargento Álvaro Eliéder, pela paciência em tolerar minha ausência pelo companheirismo, pelas palavras de conforto e que me faz muito feliz.

Aos meus colegas de faculdade por todos os momentos vividos juntos, as amigas Viviane Boll, Bruna Bergamo e Graciela Holler, pelos bons momentos, conversas, risadas e pelo incentivo.

Aos amigos e companheiros de trabalho, Everton P. Cruz, Denis Alberto Dalmolin e João Neumann Neto, eternamente grata eu serei, por me oferecerem aulas as quais foram muito importantes para o meu aprendizado e conseguir realizar este grande sonho.

Ao professor Mauro Marcelo Mattos, pela compreensão, pelo incentivo e apoio.

Em especial ao professor Wilson Pedro Carli, que me ajudou muito no decorrer desse trabalho, pelo incentivo e apoio.

Ao meu orientador, Ricardo Alencar de Azambuja, pelo incentivo, simpatia, e presteza no auxílio.

Àqueles que acreditaram no meu potencial, de concluir esta etapa. Palavras não são suficientes para exprimir o meu reconhecimento e gratidão. Peço a Deus, que ilumine a vida de cada um, hoje e sempre,

Muito Obrigada!

Tudo o que um sonho precisa para ser realizado é alguém que acredite que ele possa ser realizado.

Roberto Shinyashiki

RESUMO

A computação móvel está em constante evolução e vem sendo considerada a tecnologia da próxima geração, inovando o conceito da interação humana com os computadores. Ela traz vantagens como a diminuição do tempo desprendido, a melhoria da troca de informações, segurança, agilidade. Este trabalho consiste no desenvolvimento de um protótipo de aplicação para reserva de veículos podendo ser executado em diversas configurações de celulares e *Personal Digital Assistants* (PDAs). Acredita-se que, com a disponibilização de um sistema de reserva de veículos, via computação móvel, serão solucionados alguns entraves, como congestionamento das centrais telefônicas, deslocamentos até os escritórios de locadoras, e o acesso ao recurso pelas pessoas fora do domicílio que não têm acesso a um computador.

Palavras-chave: Computação móvel. Reserva de veículos. PDAs.

ABSTRACT

Mobile computing is evolving and is considered the next generation technology, innovating the concept of human interaction with computers. She brings advantages such as: the decrease detachable time improving the exchange of information, security, agility. This work consists in the development of a prototype application for reservation of vehicles which run on various configurations of mobile phones and personal digital assistants (PDAs). It is believed that, with the provision of a system of reserve vehicles, via mobile computing, will be resolved some obstacles, such as congestion of telephone exchanges, offsets up rental offices, access to the resource by people outside of the home that do not have access to a computer.

Key-words: Mobile computing. Reserve vehicles. PDAs.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

API – *Application Programming Interface*

CDC – *Connected Device Configuration*

CLCD – *Connected Limited Device Configuration*

EJB – *Enterprise Java Beans*

HTTP – *Hypertext Transfer Protocol*

J2EE – *Java 2 Enterprise Edition*

J2ME – *Java 2 Micro Edition*

J2SE – *Java 2 Standart Edition*

JAD – *Java Application Descriptor*

JAR – *Java Archive*

JVM – *Java Virtual Machine*

KVM – *Kilo Virtual Machine*

MIDP – *Mobile Information Device Profile*

PC – *Personal Computer*

PDA – *Personal Digital Assistant*

RMS – *Record Management System*

SOAP – *Simple Object Access Protocol*

UML – *Unified Modeling Language*

XML – *eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

A telefonia celular vem se expandindo no mundo em taxas crescentes e, tornou-se imprescindível tanto no ambiente de trabalho quanto na vida privada. De acordo com Teixeira (2005, p. 1), há dois bilhões de celulares no mundo, marca alcançada em setembro de 2005. Segundo Alencar (2006, p. 1), a base de celulares no Brasil alcançou a marca de 89,4 milhões em março de 2006. Hoje o aparelho celular é bastante diferente de 10 anos atrás, conforme Menezes (2003, p. 2), pois além de celular é também máquina fotográfica, *Mpeg Layer* 3 (MP3) *Player*, *Personal Digital Assistant* (PDA), rádio, internet, executor de aplicativos.

Segundo Almeida (2004, p. 21), para a execução de aplicativos dentro dos celulares, é necessário algo para desenvolver os aplicativos. A tecnologia Java possui a edição *Java 2 Micro Edition* (J2ME) para pequenos dispositivos como *pagers*, telefones celulares, *set-top boxes* de TVs a cabo, PDA, sendo uma versão específica da máquina virtual criada para ser executada em um ambiente com recursos limitados de memória e processamento. Os desenvolvedores são livres para criar aplicações e executá-las em qualquer dispositivo de qualquer fabricante que possua uma máquina virtual, não sendo necessário se prender a um dos fabricantes ou a uma tecnologia. Também é livre a escolha do modelo de telefone celular que possa executar o aplicativo desejado.

Desta forma, este trabalho consiste na construção de um protótipo de aplicação móvel, para reserva de veículos, acessível a partir de um aparelho celular com acesso a internet, que atenda a especificação Java J2ME. Partindo da premissa de que quem precisa de um veículo de aluguel está fora de seu domicílio, geralmente em viagem, e nem sempre conhece a localização dos serviços do local, o que gera transtorno na hora de locar qualquer bem, inclusive um veículo, aponta-se a necessidade de apresentar a possibilidade de locar este veículo em ambiente móvel, disponibilizando esse serviço em qualquer momento e lugar onde o usuário necessitar dele.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é apresentar um protótipo de aplicação móvel para reserva de veículos, acessível a partir de um aparelho celular, e que atenda a especificação Java J2ME. Os objetivos específicos do trabalho são:

a) desenvolver um sistema que permita ao cliente efetuar a reserva de veículos junto à empresa locadora utilizando-se de conexões com o servidor da mesma;

b) executar a aplicação no telefone celular interagindo com um servidor de aplicações;

c) efetuar o sincronismo entre os dados registrados no celular e os dados residentes em um servidor;

1.2 ESTRUTURA DO TRABALHO

Este trabalho está disposto em quatro capítulos.

No primeiro capítulo apresenta-se a introdução, os objetivos e a estrutura do trabalho.

No segundo capítulo tem-se a fundamentação teórica, destacando-se os conceitos de cada elemento envolvido neste processo, bem como trabalhos correlatos.

No terceiro capítulo é apresentado o desenvolvimento do aplicativo, incluindo detalhes sobre a especificação, implementação e tecnologia utilizada.

No quarto capítulo apresenta-se a conclusão sobre o trabalho, enfatizando os objetivos alcançados, bem como sugestões para trabalhos futuros como extensão deste.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo apresenta os principais temas relacionados com este trabalho de conclusão de curso. São apresentados assuntos como dispositivos móveis, plataforma JAVA, J2ME, configuração CLDC, perfil MIDP, bibliotecas kSOAP2 e kXML, *Web Service* e trabalhos correlatos.

2.1 DISPOSITIVOS MÓVEIS

Segundo Laudon e Laudon (1999, p. 160), novas formas de comunicações estão sendo proporcionadas por redes móveis de dados, usando dispositivos móveis, como telefones celulares e PDAs, um computador manual composto de uma caneta, com recursos internos e organizacionais de comunicações. Muito mais do que assistentes pessoais ou agendas eletrônicas, os dispositivos móveis passaram a ser computadores que podem ser facilmente levados a qualquer lugar, criados para atender profissionais e pessoas em movimento que necessitam de rapidez, facilidade e segurança no acesso a informações corporativas e pessoais.

Segundo Pamplona (2005 p. 17), a computação móvel é caracterizada por um dispositivo móvel com capacidade de processamento em um ambiente sem fio. Exemplos destes equipamentos são os PDAs, Pocket PCs, Smartphones e celulares, tais como os apresentados na figura 1.

Fonte: Yoshimura (2006).

Figura 1 - Exemplos de dispositivos móveis

As inovações trazidas pela tecnologia *wireless* fizeram com que a indústria deste setor tenha tido um crescimento explosivo nos últimos anos, permitindo que as pessoas comuniquem-se de forma barata e fácil sem ficarem presas aos seus telefones ou computadores de mesa.

2.2 PLATAFORMA JAVA

Segundo Montenegro e Pereira (2005, p. 28), as aplicações Java existentes englobam as variadas áreas do conhecimento e rodam nas mais diversas plataformas e sistemas operacionais, desde equipamentos bastante limitados (como celulares, PDAs, computadores de bordo) até poderosos clusters de servidores, atendendo a centenas ou milhares de clientes simultâneos. A plataforma Java foi dividida em três edições:

a) *Java 2 Standard Edition* (J2SE): o núcleo da plataforma, com a máquina virtual e as *Application Programming Interface* (APIs) básicas;

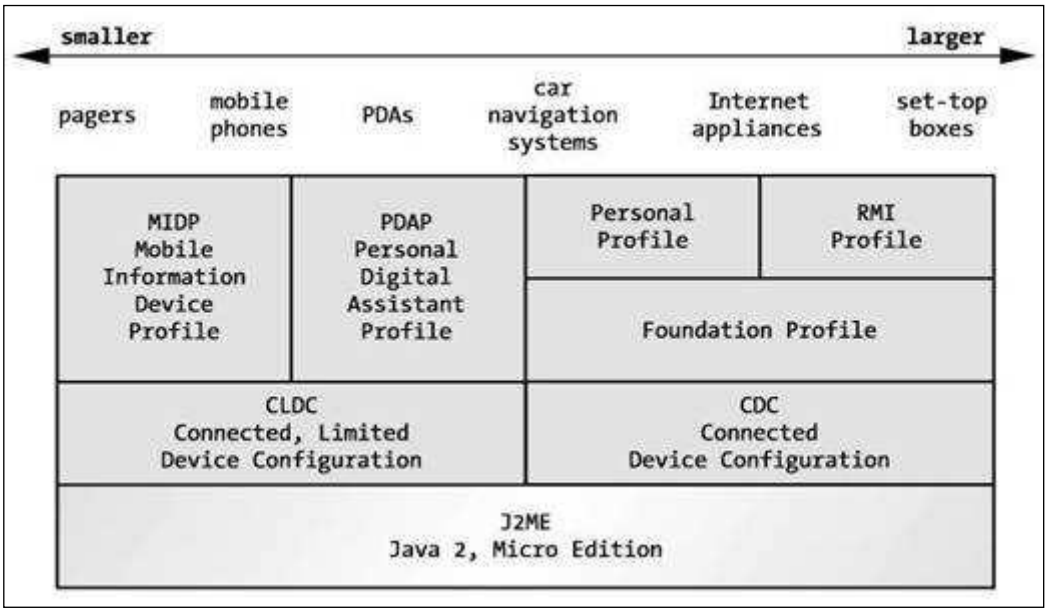
b) *Java 2 Enterprise Edition* (J2EE): complementando a J2SE e fornecendo recursos para o desenvolvimento de aplicações *Web* e corporativos;

c) *Java 2 Micro Edition* (J2ME): definindo um ambiente Java para dispositivos móveis.

2.3 J2ME

De acordo com Fonseca (2002), o Java 2 *Plataform Micro Edition* (J2ME) na verdade é um conjunto de especificações que tem por objetivo disponibilizar uma JVM, API e ferramentas para equipamentos portáteis e qualquer dispositivo com poder de processamento menor que os atuais computadores de mesa. A plataforma J2ME oferece para tanto uma máquina virtual Java, chamada de KVM, pequena o bastante para ser suportada dentro das restrições de memória destes dispositivos com memória, display e capacidade de energia reduzida.

Conforme Fonseca (2002, p. 07), a Máquina Virtual Java do J2ME, normalmente chamada de *Kilobyte Virtual Machine* (KVM), a qual é implementada de acordo com a especificação da comunidade Java, não é a mesma utilizada pelas versões J2EE e J2SE, sendo quase sempre um conjunto menor dessas versões. Por isso, códigos não podem ser portados diretamente de uma versão maior de Java para o J2ME. A relação entre os dispositivos móveis e J2ME está representada na Figura 2.



Fonte: Knudsen (2003, p. 2)

Figura 2 – Dispositivos Móveis e J2ME

O *Profile*, ou perfil, define um conjunto de bibliotecas específicas para classes de dispositivos. Segundo Corbera (2005), a diferença entre Configuração e Perfil é que a Configuração descreve de forma geral uma família de dispositivos, enquanto o Perfil fica

mais específico para um tipo particular de aparelho em uma dada família. O Perfil tão somente acrescenta funcionalidades àquele aparelho.

A configuração é um conjunto de bibliotecas básicas disponíveis para o programador. Ela também define qual o nível de serviços e funcionalidades oferecidos pela máquina virtual. Uma é a *Connected Limited Device Configuration* (CLDC), que fornece um conjunto de Java API para aplicações sem fio, ou seja, que sejam suportadas pelo dispositivo móvel. Essa especificação fornece as classes responsáveis pela conexão, entrada e saída de dados, classes de manipulações de strings e de operações matemáticas. A outra é a *Mobile Information Device Profile* (MIDP), que oferece uma biblioteca de interface gráfica para o dispositivo móvel. Essa especificação provê ainda as classes para memória persistente e algumas classes que definem objetos de formulário. Além das configurações e perfis, ainda existem as bibliotecas chamadas pacotes opcionais, que são bibliotecas de programação específicas a uma determinada tecnologia. Elas aumentam a capacidade do ambiente, caso estejam implementadas no dispositivo.

Uma configuração obrigatoriamente representa as mínimas características de uma plataforma para um dispositivo (TOPLEY, 2002). Com isso, os fabricantes são obrigados a implementar por total a especificação de uma configuração almejada para tal dispositivo. Isso garante que programadores possam desenvolver aplicações sem se preocupar em qual dispositivo específico elas irão rodar, apenas precisam da certeza que o dispositivo implementa a configuração escolhida. Há duas configurações disponíveis para a utilização do J2ME, a *Connected Device Configuration* (CDC), voltada para dispositivos com maior poder operacional e recursos de memória e conectividade, e a *Connected Limited Device Configuration* (CLDC), voltada para dispositivos com recursos limitados e que é a configuração escolhida neste trabalho.

De acordo com Topley (2002), os perfis complementam as configurações com a adição de classes que provêm características específicas para um determinado tipo de dispositivo ou um determinado segmento de aplicação, como recursos de interface gráfica, armazenamento persistente, segurança e conectividade. Cada configuração da J2ME tem um ou mais perfis associados. Segundo White (2002), alguns perfis associados à configuração CDC são:

a) *Foundation Profile*;

b) *Personal Profile*;

c) *Remote Method Invocation* (RMI) *Profile*;

d) *Personal Basis Profile*.

White e Hemphill (2002), descreve também que os seguintes perfis são referentes a configuração CLDC:

a) *Mobile Information Device Profile* (MIDP);

b) *Personal Digital Assistant Profile* (PDAP).

Os perfis *Multimedia, Gaming e Telephony Profile* atendem tanto a configuração CDC como CLDC.

2.4 CONFIGURAÇÃO CLDC

Conforme Schmitt Junior (2004), a *Connected Limited Device Configuration* (CLDC) é destinada para os dispositivos com as mais limitadas configurações do mercado, como telefones celulares, *pagers*, simples PDAs, entre outros dispositivos, que são caracterizados pelos baixos recursos de memória, processamento e conectividade. A CLDC é a configuração que provê funções de núcleo que são usadas como base para alguns perfis.

Como características principais, os dispositivos que suportam a CLDC devem ter:

a) no mínimo 192 kb de memória total disponível para a plataforma Java;

b) um processador de 16 ou 32 bits;

c) conectividade a algum tipo de rede por meio de tecnologia sem fio, conexão intermitente e com banda limitada.

Cada configuração do J2ME consiste de um conjunto de bibliotecas e de uma máquina virtual Java. A CLDC define uma máquina virtual que pode ser considerada uma versão simplificada da *Java Virtual Machine* (JVM) utilizada na edição J2SE. A Sun especifica

como deve trabalhar uma máquina virtual para a configuração CLDC e também oferece uma máquina virtual junto ao kit de desenvolvimento para J2ME chamada *Kilobyte Virtual Machine* (KVM). Porém, a utilização da KVM em aplicações J2ME não é obrigatória, podendo assim cada fabricante de dispositivos móveis implementar a sua máquina virtual que atenda à especificação (SCHMITT JUNIOR, 2004).

2.5 PERFIL MIDP

O perfil *Mobile Information Device Profile* (MIDP) trabalha sobre a configuração CLDC e habilita recursos de conectividade, armazenamento de dados local e interface gráfica com usuário. Por padrão, o perfil MIDP carece de recursos avançados de segurança e interface gráfica, porém os fabricantes dos dispositivos podem oferecer recursos opcionais que customizam essas carências. Uma aplicação que implementa o perfil MIDP é chamada de MIDlet e roda sobre a máquina virtual KVM proposta pela configuração CLDC. Um MIDlet pode utilizar tanto funções oferecidas no perfil MIDP como funções que o MIDP herda da CLDC (SCHMITT JUNIOR, 2004).

Os dispositivos atendidos pelo perfil MIDP são normalmente telefones celulares e PDAs. De acordo com a especificação deste perfil, o dispositivo deve ter:

a) 128 kb de memória não volátil necessária para armazenamento da implementação;

b) 32 kb de memória volátil para a pilha Java;

c) 8 kb de memória não volátil necessária para armazenamento local persistente;

d) visor de pelo menos 96 x 54 pixels;

e) algum recurso para entrada de dados como teclado, tela por toque ou área de teclas conforme telefones celulares;

f) possibilidade de conexão a algum tipo de rede.

Uma aplicação MIDP contém no mínimo uma classe MIDlet. Porém pode haver casos em que uma aplicação contém mais de um MIDlet, e neste caso a aplicação é chamada de MIDlet Suite. Ao ser carregada no dispositivo, a MIDlet Suite, faz a varredura de todos os MIDlets existentes e dispõe ao usuário escolher qual MIDlet lançar. A aplicação MIDP, para ser instalada no dispositivo, precisa ser empacotada, construindo um *Java Archive* (JAR). O JAR obrigatoriamente contém um arquivo de manifesto que engloba informações sobre o MIDlet ou conjunto de MIDlets contidos no pacote JAR. O JAR trabalha em conjunto com um *Java Application Descriptor* (JAD). O descritor contém informações sobre a aplicação, principalmente sobre a configuração e perfil utilizados pela mesma (SCHMITT JUNIOR, 2004).

2.6 *WEB SERVICES*

Gumz (2005) resume o *Web Service* como um padrão de computação distribuída, na qual deve existir a criação, publicação, localização e acesso por sistemas remotos. O *Web Service* pode ser visto como um serviço, publicado na forma de um componente de software independente de implementação e plataforma, onde suas interfaces públicas e regras são definidas e descritas usando o XML. O *Web Service* não precisa necessariamente estar disponível somente no ambiente *Web*, pois o mesmo pode ser utilizado em qualquer rede de trabalho, Internet ou Intranet, onde geralmente é utilizado o protocolo *Hypertext Transfer Protocol* (HTTP) para efetuar a comunicação entre o cliente e o *Web Service*. Para se obter a integração e a portabilidade entre as aplicações de uma organização ou entre parceiros de negócio, é utilizado a tecnologia XML, componente chave do *Web Service*, que possibilita que ambas as partes, cliente e provedor do serviço, troquem mensagens com informações, mesmo estando em ambientes e sistemas diferentes.

2.7 BIBLIOTECA kSOAP2 E kXML

Segundo Rosa (2005, p. 66), kSOAP2 e kXML são bibliotecas para a utilização de SOAP e XML otimizadas para serem executadas em uma aplicação J2ME. Estas bibliotecas foram utilizadas no sistema a fim de oferecer a comunicação do dispositivo móvel com o *Web Service*, decodificando envelopes SOAP no formato XML. O apêndice B demonstra a utilização da biblioteca kSOAP2.

2.8 TRABALHOS CORRELATOS

Schmitt Junior (2004) desenvolveu um protótipo de *front end* de Controle de Acesso, utilizando J2ME, onde tem como objetivo principal automatizar o controle de segurança patrimonial, oferecendo flexibilidade na aplicação, em relação ao modo de armazenamento das informações e a interface com o usuário, se beneficiando com os recursos de um dispositivo móvel, baseado na tecnologia J2ME. No desenvolvimento do trabalho, são explorados recursos avançados do J2ME, como por exemplo, o *Record Management System* (RMS), para o armazenamento das informações no dispositivo e a troca de informações com base no protocolo HTTP, utilizando o formato XML para a troca de dados com um servidor.

Rosa (2005), desenvolveu um protótipo de uma aplicação para dispositivos móveis onde, também como neste trabalho, foi utilizada a tecnologia J2ME, possuindo como base o diário de classe da FURB, que foi implementado para utilizá-lo no telefone celular. O módulo do telefone celular também se comunicou com um *Web Service* em um servidor de aplicações, via *Hyper Text Transfer Protocol Secure* (HTTPS) e SOAP.

3 DESENVOLVIMENTO

Este capítulo tem por objetivo demonstrar as fases executadas para concepção, análise, e desenvolvimento do protótipo, ou seja, a análise dos requisitos contemplando os requisitos funcionais e não funcionais, a especificação contendo os principais casos de uso, diagrama de atividades, MER, seguidos pelo desenvolvimento (implementação) e dos resultados obtidos.

3.1 LEVANTAMENTO DE INFORMAÇÕES

Com o crescimento na utilização dos serviços de locações de veículo e através de algumas pesquisas informais, chegou-se a conclusão de certa carência de um produto adequado, onde se identifica a necessidade de oferecer uma nova opção para pessoas que constantemente fazem uso deste serviço, há agora uma tecnologia que traz segurança e mobilidade superando o atual sistema onde o Cliente precisava se deslocar até a agência de locação para efetuar uma locação de veículo.

Sendo assim, chega-se a um projeto inovador, oferecendo possibilidades ainda não utilizadas em grande escala para atender um novo mercado de clientes que procuram mobilidade, agilidade, facilidades e segurança em seu cotidiano, ao qual este protótipo visa atender.

O software é composto por três módulos:

a) módulo MIDP MarciRentCliente que é executado em um aparelho celular, utilizado pelo cliente, o qual utiliza a tecnologia J2ME, sendo que este módulo se comunica com um *Web Service* em um servidor de aplicações , via HTTP e SOAP;

b) módulo ReservaCar que é executado na *Web*, o qual utiliza a tecnologia JSP, onde são mantidas todas as informações dos veículos, clientes e reservas no banco de dados;

c) *Web Service* para possibilitar a comunicação do módulo MIDP MarciRentCliente

com o módulo ReservaCar.

A Figura 3 demonstra a comunicação entre os módulos apresentados ida e volta das informações.

Figura 3 – Comunicação entre módulos

3.2 ESPECIFICAÇÃO

Os requisitos, classificados como Requisitos Funcionais (RF) e Requisitos Não Funcionais (RNF) descrevem o que o sistema deve e o que não deve fazer. Os RF apresentam as funcionalidades e o comportamento que o sistema deve possuir em determinadas situações. Os RNF apresentam as restrições que o sistema terá sobre alguns serviços ou funções oferecidas como usabilidade, navegabilidade, portabilidade, segurança e *hardware*.

Os requisitos de um software são as descrições sobre seu comportamento, restrições das operações que deve realizar e especificações sobre suas propriedades ou atributos. Os requisitos compreendem as funcionalidades presentes no software quando este estiver pronto para ser executado (KOSCIANSKI; SOARES, 2006, p.174).

3.2.1 REQUISITOS FUNCIONAIS

Com base na idéia proposta de construção de um protótipo de aplicação móvel responsável por permitir efetuar uma reserva de veículo a partir de um dispositivo móvel a seguir são demonstrados os levantamentos dos requisitos funcionais e requisitos não funcionais estabelecidos pela análise final.

O Quadro 1 apresenta os requisitos funcionais previstos para o sistema e sua rastreabilidade, ou seja, vinculação com o(s) caso(s) de uso associado(s).

Requisitos Funcionais		Caso de Uso
RF01	O sistema deverá possibilitar a reserva do veículo pelo Cliente	UC01
RF02	O sistema deverá permitir ao Cliente visualizar os dados da reserva	UC02
RF03	O sistema de retaguarda deverá permitir incluir os veículos para locação	UC03
RF04	O sistema de retaguarda deverá permitir alterar/excluir os veículos para locação	UC04
RF05	O sistema de retaguarda deverá permitir incluir os clientes	UC05
RF06	O sistema de retaguarda deverá permitir alterar/excluir os clientes	UC06
RF07	O sistema de retaguarda deverá permitir efetuar/consultar reservas	UC07

Quadro 1: Requisitos funcionais

3.2.2 REQUISITOS NÃO FUNCIONAIS

O Quadro 2 lista os requisitos não funcionais previstos para o sistema.

Requisitos Não Funcionais	
RNF01	A comunicação entre o módulo ReservaCar e o módulo MIDP MarciRentCar deve ser através de *Web Service* com SOAP
RNF02	A interface com usuários do sistema será através do dispositivo celular e browser *Web*
RNF03	Utilizar o banco de dados MySQL
RNF04	Deve ter compatibilidade com recursos do perfil MIDP 2.1 do J2ME
RNF05	Deve ter compatibilidade com a plataforma CLDC

Quadro 2: Requisitos não funcionais

3.2.3 CASO DE USO

De acordo com Souza e Lima-Cardoso (2007, p.46), um *use-case* demonstra uma unidade da funcionalidade provida pelo sistema. A funcionalidade principal deste diagrama é ajudar as equipes do desenvolvimento a visualizar as exigências funcionais que o software

deverá contemplar, considerando inclusive o relacionamento dos “atores”, ou seja, pessoas que irão interagir com os programas, com os processos essenciais.

Esta seção apresenta o diagrama de casos de uso, sendo que os detalhamentos dos principais casos de uso são descritos no Apêndice A. Na figura 4 tem-se o diagrama de casos de uso.

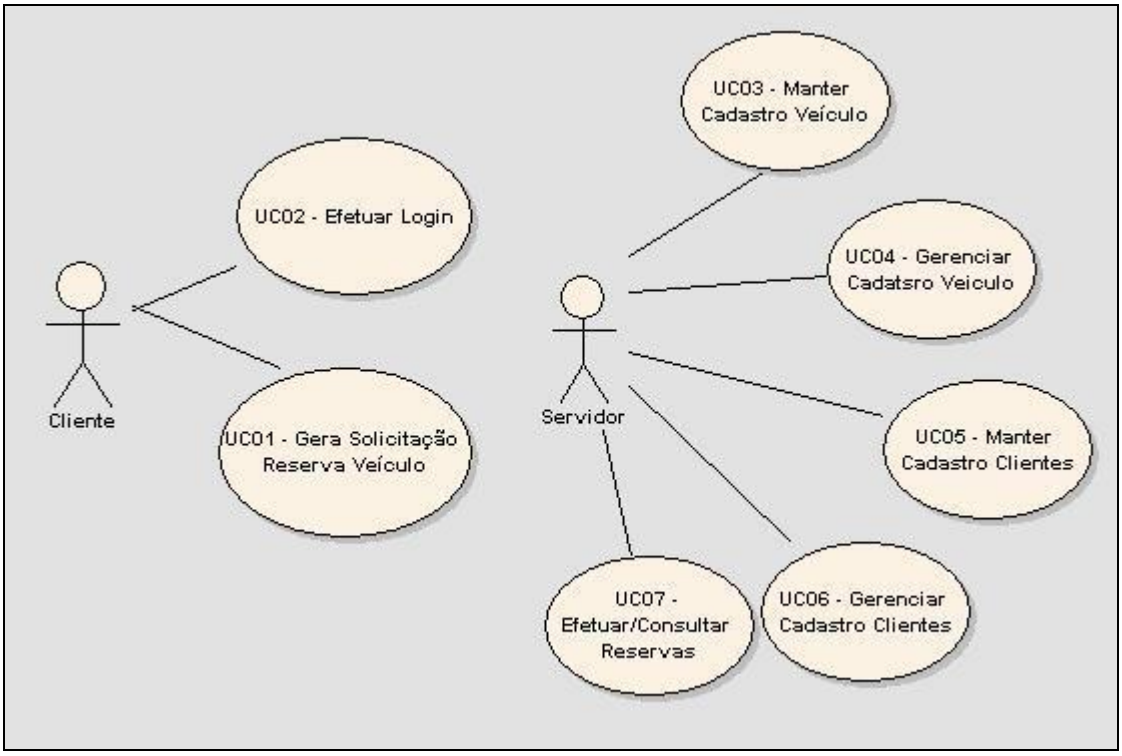


Figura 4 - Caso de Uso

3.2.4 MODELO ENTIDADE-RELACIONAMENTO

Na figura 5 se apresenta o modelo entidade-relacionamento onde estão às tabelas que são persistidas no banco de dados referente à aplicação servidor.

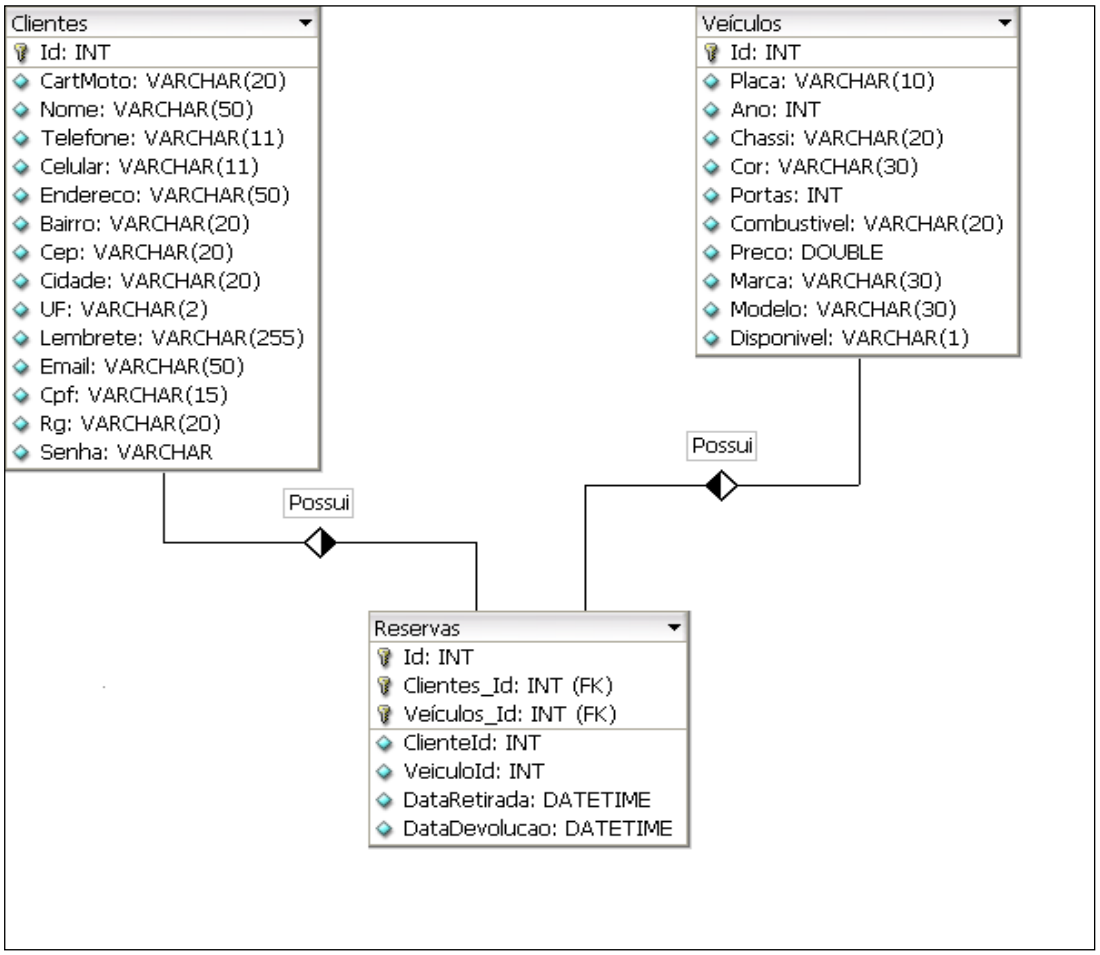


Figura 5 – Modelo Entidade-Relacionamento

A seguir é apresentada uma breve descrição das entidades utilizadas para o desenvolvimento do sistema:

a) clientes: entidade responsável por armazenar informações referentes aos clientes do sistema servidor;

b) reservas: entidade responsável por armazenar informações referentes às reservas dos clientes;

c) veículos: entidade responsável por armazenar informações referentes aos veículos.

3.2.5 DIAGRAMA DE ATIVIDADES

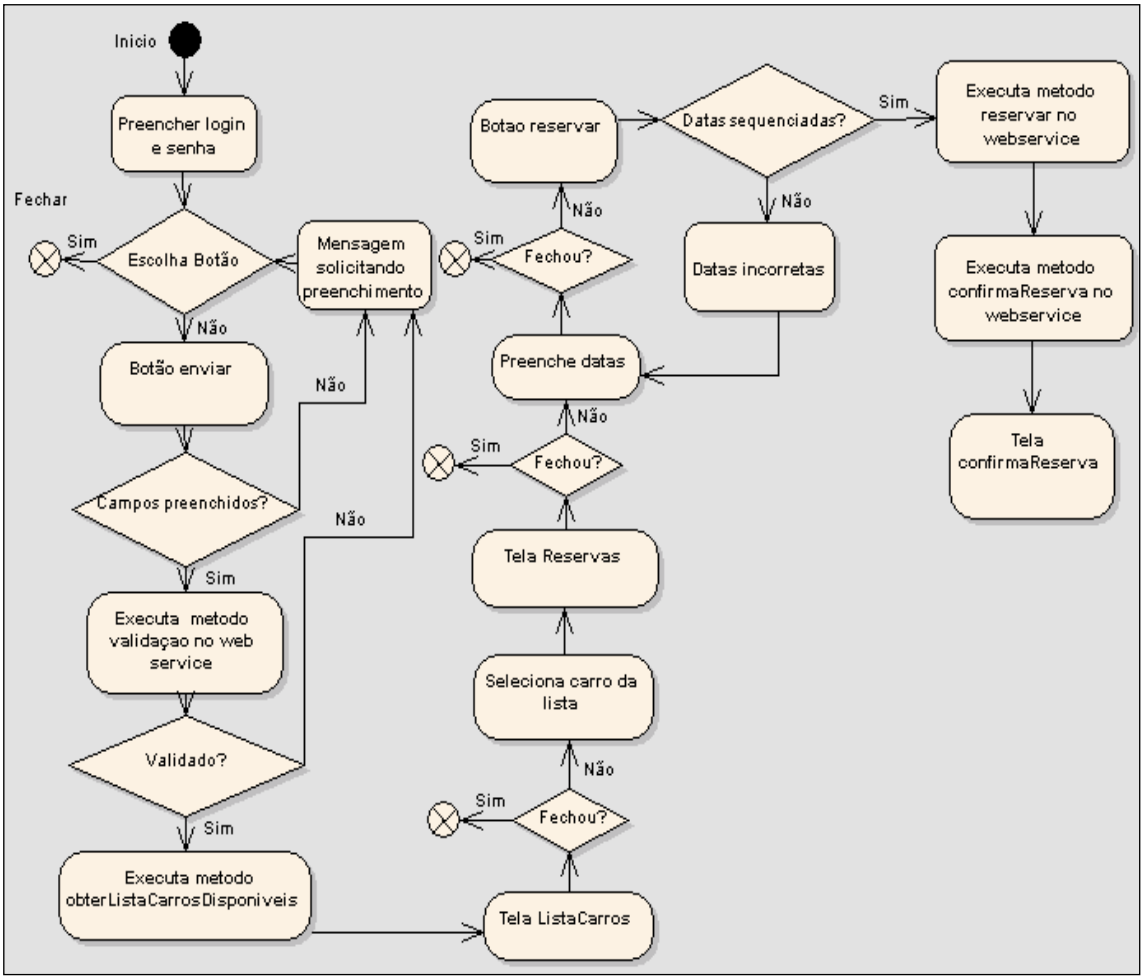


Figura 6 – Diagrama de atividades aplicação cliente

A figura 6 mostra o diagrama de atividades do caso de uso relacionado ao cliente, desde a solicitação de usuário e senha no dispositivo móvel até a confirmação ou não da reserva de um veículo.

3.3 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas na aplicação cliente e servidor.

3.3.1 TÉCNICAS E FERRAMENTAS UTILIZADAS

O módulo servidor foi desenvolvido utilizando a linguagem JSP fazendo conexão com banco de dados MYSQL, realizando tratamentos e validação via Java Script. Também usou-se a tecnologia AJAX no formulário da reserva do veículo. No módulo cliente aplicou-se a linguagem Java para dispositivos móveis (J2ME) com perfil MIDP 2.1, ambos os módulos através da IDE Netbeans 6.9.1 onde o código foi escrito, compilado e depurado.

A Plataforma Netbeans é uma base genérica para aplicativos de área de trabalho. Ela fornece os serviços comuns para quase todos os grandes aplicativos de áreas de trabalho: gerenciamento de janelas, menus, configurações e armazenamento, um gerenciador de atualizações e acesso a arquivos. A estrutura da IDE pode ser vista na figura 7.

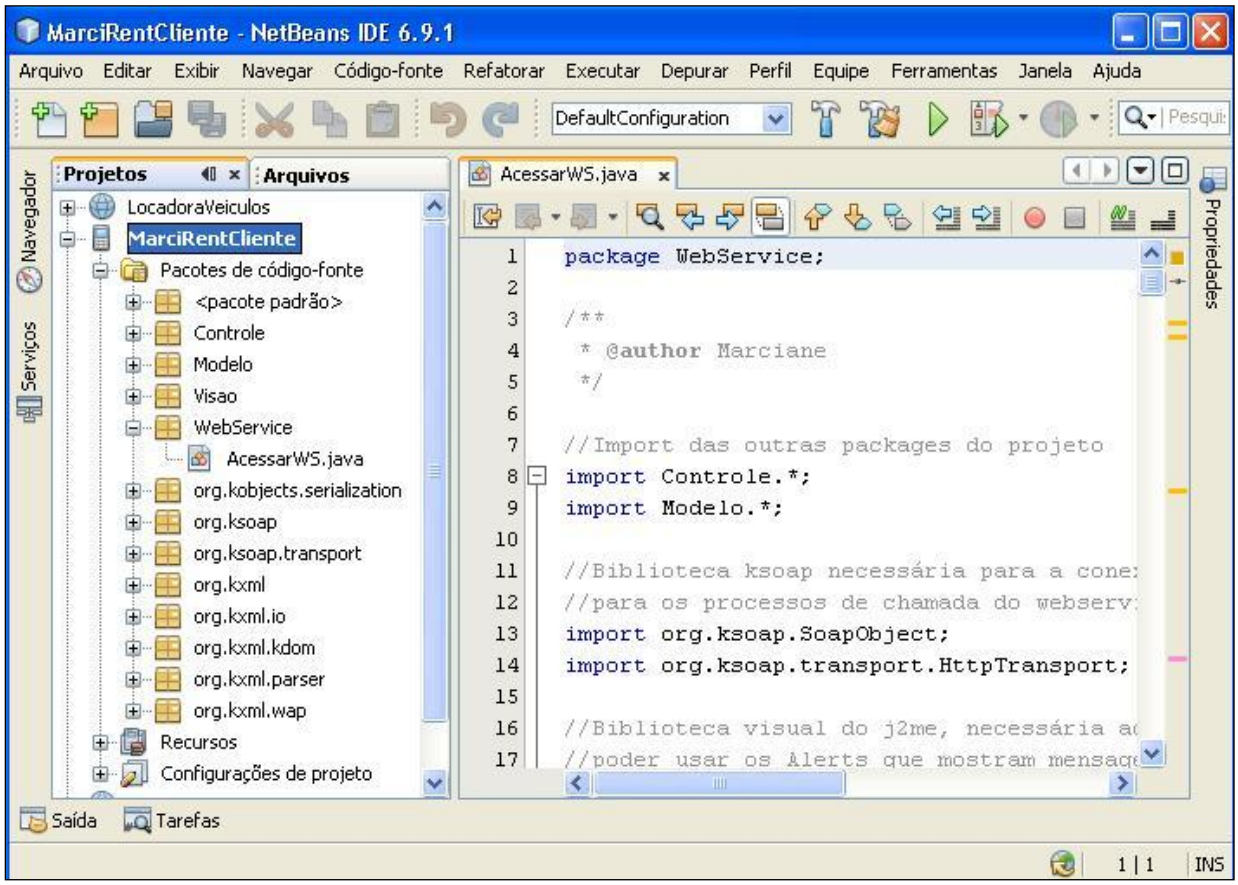


Figura 7 – IDE Netbeans

Abaixo cita-se algumas ferramentas utilizadas:

a) Netbeans, versão: 6.9.1 ferramenta utilizada para desenvolvimento nos serviços *web* e na aplicação para celulares;

b) SQL Server, versão: 5.5 base de dados de informações do sistema;

c) HeidiSQL, versão: 6.0.0.3603 ferramenta utilizada para gerenciar o banco de dados Mysql;

d) Enterprise Architect, versão: 6.5 ferramenta utilizada para criação do diagrama de atividades e casos de uso;

e) DB Designer, versão: 1.0.42 ferramenta utilizada para criação do modelo entidade relacionamento;

f) Tomcat versão: 6.0 por ser um servidor *web* Java foi utilizado na aplicação pelo fato da mesma ser quase que totalmente baseada em Java Server Pages (JSP);

e) AXIS versão: 1.4 o Apache Axis por ser um framework de código aberto, baseado na linguagem Java e no padrão XML, foi utilizado para a construção do *Web Wervice* no padrão SOAP.

3.3.2 OPERACIONALIDADE DA IMPLEMENTAÇÃO

A seguir serão demonstradas as telas do módulo ReservaCar e do módulo MIDP MarciRentCliente, com o passo a passo da utilização de cada módulo.

3.3.2.1 MÓDULO RESERVACAR

Este módulo do sistema é utilizado pelos funcionários, através da *Web* a fim de disponibilizar as informações ao Cliente no momento da locação. Na Figura 8 é demonstrada a interface *web* na qual o cliente irá autenticar-se com seu nome de usuário e senha para ter acesso ao sistema servidor.

Figura 8 – Tela de login do usuário na interface aplicação *web*

O sistema só vai autenticar se utilizar um usuário e senha que esteja no banco de dados, na tabela de funcionários, ao digitar usuário e senha inválidos, o usuário poderá visualizar a mensagem. Após autenticar, será apresentada a tela de menu conforme figura 9.

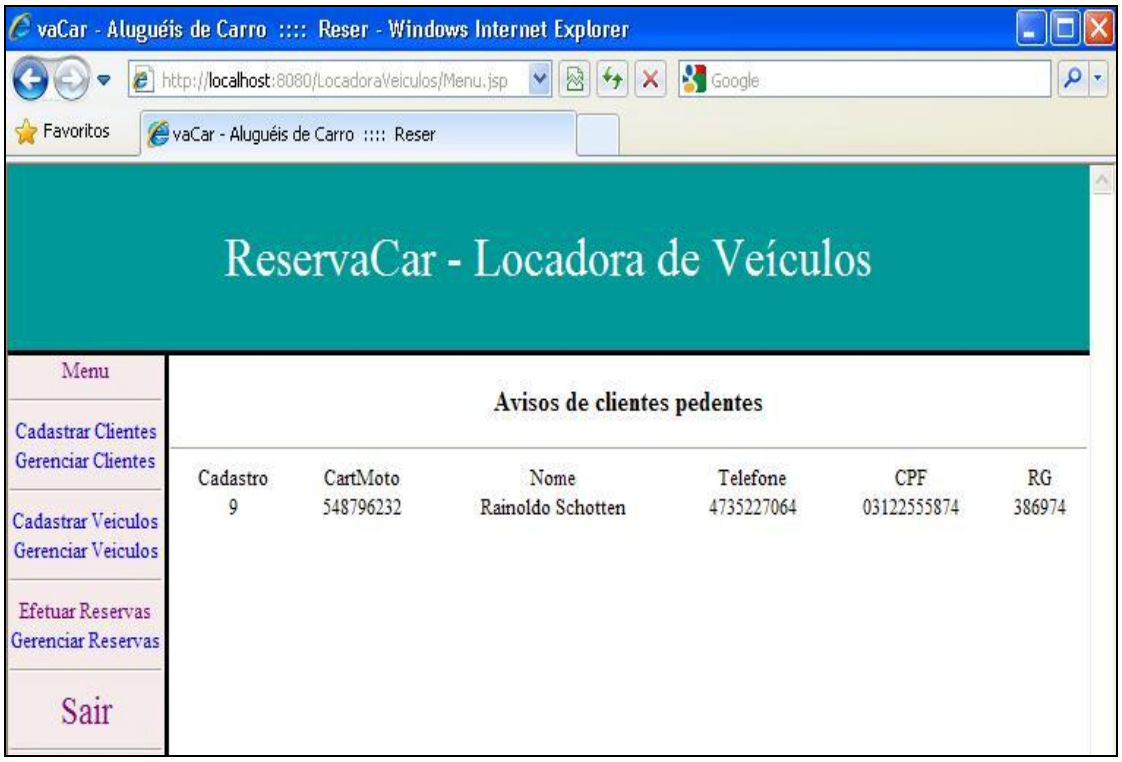


Figura 9 – Tela de menu da aplicação *web*

Pode-se observar, no exemplo da figura 9, um aviso referente a clientes pendentes na data corrente, ou seja, o cliente Rainoldo Schotten, com cadastro de número 9, possui uma reserva e, o funcionário que acessar a aplicação, vai receber esse alerta com todos os clientes que possuem reserva, onde a data para devolução é a data corrente.

Acessando a opção Cadastrar Clientes, o sistema dispõe dos campos para preenchimento com os dados do cliente bem como o botão salvar e voltar. Campos obrigatórios são destacados com asterisco, ao preencher os campos e optar pelo botão salvar, será apresentada a mensagem “deseja inserir novo registro?”. Confirmando, vai permanecer na tela cadastro de clientes, senão, será redirecionado para a tela menu. Na figura 10 temos um exemplo de confirmação de cadastro.

Conforme a tela do menu representada na figura 9, há ainda a opção de cadastrar veículo e gerenciar veículo, semelhante ao apresentado anteriormente. A consulta de veículos foi feita pela placa e chassi.

Figura 10 – Tela do cadastro de clientes

Acessando o menu Gerenciar Clientes, conforme figura 11, está disponível o campo de buscas, onde o sistema permite efetuar consultas por nome do cliente e pela carteira de motorista. Se não for selecionada opção de busca, clicando apenas em buscar, todos os clientes que estão gravados no banco de dados na tabela de Clientes serão listados, juntamente com seus dados, apresentando ainda, no final da listagem a quantidade de clientes cadastrados. Com um ou mais clientes em tela, pode-se optar pelos botões Apagar ou Atualizar, selecionando um ou mais clientes, optando pelo botão Apagar, será excluído do banco de dados, apresentando a mensagem “Cliente(s) com cadastro(s) x, y, z excluído(s) com sucesso”. Se a opção escolhida for Atualizar, os dados do cliente selecionado serão apresentados na tela podendo ser alterados e gravados clicando na opção Salvar. Na atualização, caso o funcionário selecione mais de um cliente, o sistema apresentará a mensagem. “Por favor, escolha somente um cadastro para a atualização”.

A tela de consulta clientes, de acordo com a figura 11, a opção de filtro utilizada foi o nome, observe que a quantidade de registros encontrados foi um.

Figura 11 – Tela de consulta de clientes

No menu Efetuar Reservas, conforme figura 12, há os campos Cliente, Veículos, Data Retirada e Data Devolução. No campo cliente, conforme for digitando o nome o sistema vai dispondo da caixa para seleção do cliente. Na caixa de seleção dos veículos, somente serão apresentados os veículos que estiverem disponíveis no banco de dados, ou seja, os veículos que não possuem reservas. Os veículos que estiverem disponíveis, é possível visualizar o modelo, marca, ano, cor, combustível, quantidade de portas e o preço por km rodado. Após fazer as escolhas acima, é preciso informar a data de retirada a data de devolução, e por fim, confirmar a reserva do veículo. Na figura 12, apresenta-se a tela referente à reserva de veículos.

Figura 12 – Tela de reserva de veículos

Clicando em Gerenciar Reservas, disponibiliza-se filtros onde o funcionário tem opções de pesquisar reservas pelo nome do cliente, modelo do veículo, data de retirada e data de devolução. Na figura 13, efetua-se uma consulta de reserva pela data de retirada, observando que o sistema além de mostrar alguns dados do cliente, lista as informações referentes ao carro que reservado.

Figura 13 – Tela de consulta reserva de veículos

Para encerrar atividades com o sistema, utiliza-se o botão Sair, ao clicar neste, será direcionado para a tela de *login* do sistema.

3.3.2.2 MÓDULO MIDP MARCIRENTCLIENT

Este módulo do sistema é utilizado pelo cliente, através do celular, por telas de informações necessárias para a locação do veículo desejado.

No quadro 3 apresenta-se parte do código fonte onde utiliza-se SOAP para conexão com o *Web Service*. O código foi totalmente comentado.

```
//Biblioteca ksoap necessária para a conexão com a net e
//para os processos de chamada do webservice.
import org.ksoap.SoapObject;
import org.ksoap.transport.HttpTransport;

//Biblioteca visual do j2me, necessária aqui apenas para
//poder usar os Alerts que mostram mensagens de erro.
import javax.microedition.lcdui.*;

public class AcessarWS {

    //Atributo necessário para manter a ligação com a classe principal
de controle.
    private MarciRentClient main;

    //String com a url do endereço base do webservice.
    private                 String                 url                 =
"http://localhost:8080/MarciRentWS/services/MarciRentWS";

    //Lista vetor para armazenar os carros que estiverem disponiveis
para locação.
    private java.util.Vector vCarros;

    public AcessarWS(MarciRentClient _main){
        //guarda uma cópia da classe controle para poder executar seus
métodos.
        main = _main;
        //Inicializa o vetor de carros
        vCarros = new java.util.Vector();
    }
```

Quadro 3 – Classe AcessarWS

No quadro 4, apresenta-se parte do código fonte com o primeiro método importante, que acesso o *Web Service*, mandando nome de usuário e senha para que sejam autenticados e autorizados.

```
public boolean autenticarUsuario(String user, String pwd){

boolean ret = false;

//Cria um objeto SOAP (XML) usando a url e o nome do método a chamar.
SoapObject client = new SoapObject(url, "autenticarUsuario");

/*Adiciona propriedades (parâmetros) conforme o método aguarda
* É muito importante aqui acompanhar os tipos dos parâmetros
* esperados. Neste caso os parâmetros são Strings, mas há métodos
* que o parâmetro precisa ser Integer e isso é muito importante
* para o sucesso da operação.*/
client.addProperty("user", user);
client.addProperty("pwd", pwd);

//Cria um objeto Http para transportar via web o objeto SOAP.
HttpTransport ht = new HttpTransport(url, "autenticarUsuario");

//Como esse método retorna um Integer (objeto, não um 'int'), é
//importante criar esse objeto de retorno .
Integer clienteID;
try{
//Faz o call usando o objeto SOAP e o transportador Http
clienteID = (Integer)(ht.call(client));

//O ws foi construído para devolver -1 se o cliente não for validado.
if(clienteID.intValue()>-1){
//Se a validação ocorreu o ID do cliente retornou e será guardado
//na classe de controle para uso do sistema.
main.setClienteID(clienteID.intValue());
ret = true;
  }
}
catch(java.io.IOException e){
//Se houver qualquer erro, o aplicativo mostra um Alert informando
//que o WebService não está disponível.
Command okCommand = new Command("Ok", Command.OK, 0);
Alert alert = new Alert("Erro", "WebService não disponivel.", null,
AlertType.ERROR);
alert.addCommand(okCommand);
main.getDisplay().setCurrent(alert);
ret = false;
}

//Retorna o sucesso, ou o fracasso da operação.
return ret;
}
```

Quadro 4 – Método de acesso ao *Web Service*

No quadro 5 apresenta-se parte do código fonte com o método para carregar uma lista com os veículos disponíveis.

```
public void carregarLista(){
SoapObject client = new SoapObject(url,
"obterListaCarrosDisponiveis");
client.addProperty("user", main.getUsuario());
client.addProperty("pwd", main.getSenha());
HttpTransport ht = new HttpTransport(url,
"obterListaCarrosDisponiveis");
//Este método retorna uma String
String s ="";
try{
s = (ht.call(client)).toString();
}
catch(java.io.IOException e){
Command okCommand = new Command("Ok", Command.OK, 0);
Alert alert = new Alert("Erro", "WebService não disponivel.", null,
AlertType.ERROR);
alert.addCommand(okCommand);
main.getDisplay().setCurrent(alert);
return;
}
```

Quadro 5 – Método com parâmetros validade do cliente

No quadro 6 apresenta-se parte do código fonte com o método "reservarVeiculo".

```
/*Este método passa ao webservice o id do carro a reservar, e os
valores das datas de retirada e devolucao*/
public boolean reservarVeiculo(int id, int diaret, int mesret, int
anoret, int diadev, int mesdev, int anodev){
boolean ret = false;
//Novo objeto SOAP com o ws e o nome do método
SoapObject client = new SoapObject(url, "reservarVeiculo");

//Construção dos parâmetros do metodo.
client.addProperty("ClienteId", new Integer(main.getClienteID()));
client.addProperty("VeiculoId", new Integer(id));
client.addProperty("diaret", new Integer(diaret));
client.addProperty("mesret", new Integer(mesret));
client.addProperty("anoret", new Integer(anoret));
client.addProperty("diadev", new Integer(diadev));
client.addProperty("mesdev", new Integer(mesdev));
client.addProperty("anodev", new Integer(anodev));

HttpTransport ht = new HttpTransport(url, "reservarVeiculo");

//Este método retorna um booleano. Optou-se por transforma-lo em
String para simplificar.
String resultado = "";
try{
//Chamada do método no ws.
resultado = (ht.call(client)).toString();
}
catch(java.io.IOException e){
Command okCommand = new Command("Ok", Command.OK, 0);
Alert alert = new Alert("Erro", "WebService não disponivel.", null,
AlertType.ERROR);
alert.addCommand(okCommand);
main.getDisplay().setCurrent(alert);
return false;
}
boolean reservado = false;
if(resultado.equalsIgnoreCase("true"))
//Se o veiculo foi reservado retorna a confirmacao.
ret =true;
return ret;
}
}
```

Quadro 6 – Método reservarVeiculo

O código do aplicativo cliente MarciRentClient, é o código normal de qualquer aplicação J2ME que cria telas e mostra informações as colocando no *Display* da classe principal e disponibilizando botões para a escolha de opções.

Este projeto está dividido em 4 *packages* o Modelo, a Visao, o Controle e o *WebService*, procurando seguir a arquitetura MVC, permitindo organizar separadamente as classes, a *package* Modelo contém apenas a classe que modela o objeto Carro. A *package* Visao possui todos os *forms* e *list* que farão a interface com o usuário e a *package* Controle possui a classe *Main* que gerencia o intercâmbio de Vistas e guarda as informações principais, a *package WebService* mantém a classe AcessarWS que contém os métodos de conexão com

a *web* e ativação dos métodos do *Web Service*.

A classe mais importante e incomum é a classe AcessarWS que faz o acesso ao *Web Service* e recupera as informações que o aplicativo vai usar em suas telas.

Para demonstrar a operacionalidade da aplicação cliente, são apresentados a seguir funções do sistema visando demonstrar um caso real da utilização.

Ao carregar a aplicação, é disponibilizado para o cliente o menu solicitando autenticação com usuário e senha conforme figura 14, os dados preenchidos devem corresponder ao campo nome e senha armazenados na tabela de clientes da aplicação *Web* ReservaCar (servidor).

Figura 14 – Tela de login

Se o usuário e senha fornecidos pelo cliente forem inválidos, será apresentada a mensagem “usuário ou senha incorretos”, observe a figura 15.

Figura 15 – Tela de falha na autenticação

Se a autenticação ocorrer com sucesso, o servidor enviará a lista dos carros disponíveis para locação, observe a figura 16. O apêndice B demonstra parte do processo de autenticação e listagem dos carros disponíveis.

Figura 16 – Tela de veículos em locação

Selecionando o carro desejado, o sistema solicita a data de retirada e a data de devolução, após, serão apresentadas duas opções, Reservar e Fechar, conforme figura 17.

Figura 17 – Tela de seleção das datas

Optando por Reservar, esses dados serão enviados corretamente para a aplicação servidora que os recebe, processa de acordo com a base de dados referente à reserva realizada e retorna a resposta para o aplicativo cliente. O módulo cliente aceita a mensagem com o resultado da reserva e mostra na tela do emulador a confirmação com os dados do veículo reservado: marca, modelo, ano, cor, placa, portas, combustível, preço por km rodado, data da retirada e a data da devolução encerrando o ciclo da reserva. Um exemplo dessa operacionalidade pode ser vista na figura 18.

Figura 18 – Tela de confirmação da reserva

Observe que permanecem disponíveis as opções, **+Reservas**, que vai iniciar todo o ciclo de locação e **Fechar** onde encerra todo processo.

3.4 RESULTADOS E DISCUSSÃO

O principal objetivo deste trabalho foi desenvolver um protótipo para reserva de veículos via dispositivo móvel. A etapa de testes e validação dos requisitos deste trabalho foi através de simulações feitas em emuladores J2ME da Sun Microsystems. A comunicação com o *Web Service* no emulador foi realizada com sucesso e, os dados são enviados corretamente para a aplicação servidora que os recebe, processa de acordo com a base de dados referente à reserva e retorna a resposta para o aplicativo cliente. O módulo cliente aceita a mensagem com o resultado da reserva e mostra na tela do emulador a confirmação com os dados do veículo reservado.

Com todos estes testes, foi possível validar os requisitos funcionais e não-funcionais implementados no trabalho com sucesso. Na aplicação servidor, foi disponibilizado filtros, possibilitando a busca pelas reservas efetuadas, com opções de pesquisas pelo nome do cliente, modelo do veículo, data de retirada e data de devolução.

O Quadro 7 relaciona os trabalhos correlatos mencionados na fundamentação teórica com a aplicação desenvolvida, evidenciando aspectos e características em cada uma delas.

Funcionalidades	Este Projeto	Rosa (2005)	Schmitt Junior (2004)
Aplicação desenvolvida com J2ME	Sim	Sim	Sim
Comunicação com *Web Services*	Sim	Sim	Não
Utilização de XML com o protocolo HTTP	Sim	Sim	Sim

Quadro 7 – Funcionalidades específicas de cada trabalho.

4 CONCLUSÕES

No presente trabalho foi constatada a evolução da comunicação móvel bem como o crescente aumento na utilização de dispositivos móveis, sendo destacados os telefones celulares. Foram realizados estudos sobre a tecnologia J2ME, seus conceitos e características. Também foram utilizados os recursos necessários para estabelecer a comunicação entre os aplicativos que rodavam em suas respectivas plataformas.

O protótipo desenvolvido nesse trabalho comprovou, através de testes realizados, ter cumprido seus objetivos, ou seja, permite o cliente acessar pelo dispositivo móvel interagindo com um servidor de aplicações, efetuando o sincronismo entre os dados registrados no celular e os dados residentes em um servidor podendo assim escolher um veículo disponível para locação confirmando uma reserva desse veículo junto à locadora. Esses dados são enviados corretamente para a aplicação servidora que os recebe, processa de acordo com a base de dados referente à reserva realizada e retorna a resposta para o aplicativo cliente. O módulo cliente aceita a mensagem com o resultado da reserva e informa na tela do emulador a confirmação com os dados do veículo reservado, marca, modelo, ano, cor, placa, portas, combustível, preço por km rodado a data da retirada e a data da devolução, encerrando o ciclo da reserva.

À medida que cresce o mercado de telefonia celular, aumenta também a diversidade de aplicativos que são desenvolvidos para esses aparelhos. Empresas de desenvolvimento criam cada vez mais softwares que trocam dados na rede e se comunicam com servidores.

4.1 EXTENSÕES

Como extensão deste trabalho, visando complementá-lo, sugere-se as seguintes extensões:

a) criar a rotina que permita a consulta, alteração ou cancelamento da reserva pelo dispositivo móvel;

b) criar a rotina que permita o cliente manter seu cadastro pelo dispositivo móvel;

c) solicitar no momento da reserva o preenchimento com os dados do cartão de crédito do cliente através de uma conexão segura;

d) criar a rotina que permita ao usuário o faturamento das locações e o controle financeiro na aplicação do servidor;

e) percebeu-se a necessidade de geração de comprovantes de reservas devidamente autenticados, boletos e relatórios de controle e estatísticas para o cliente.

REFERÊNCIAS BIBLIOGRÁFICAS

ALENCAR, P. Brasil tem 89,4 milhões de celulares. **Plantão INFO**, São Paulo, abr. 2006. Disponível em: <http://info.abril.com.br/aberto/infonews/042006/24042006-5.shl>. Acesso em: 23 out. 2010.

ALMEIDA, Leandro Batista de. et al. **Introdução à J2ME e programação MIDP**. Mundo Java, Rio de Janeiro, n. 5, p. 20-27, 2004.

CORBERA, Rodrigo G. **Tutorial de programação J2ME**. [S.l.], [2005]. Disponível em: <http://www.wirelessbrasil.org/wirelessbr/colaboradores/corbera_martins/j2me_01.html>. Acesso em: 17 abr. 2011.

FONSECA, Jorge C. **Portando a KVM**. 2002. 64 f. Trabalho de Conclusão de Curso (Graduação em Ciências da Computação) – Centro de Informática, Universidade Federal de Pernambuco, Recife. Disponível em: <http://www.cin.ufpe.br/~tg/2002-1/jcbf.doc>. Acesso em: 16 maio 2010.

GUMZ, Rafael Araújo. **Protótipo de um sistema gerador de interfaces gráficas para testes de Java Web Services**. 2005. 98 f. Monografia de Pós-Graduação (Pós-Graduação em Desenvolvimento de Aplicações para WEB) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

KOSCIANSKI, André; SOARES, Michel dos Santos. **Qualidade de software:** aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software. São Paulo: Novatec, 2006.

KNUDSEN, Jonathan. **Wireless Java:** developing with J2ME. 2.ed. New York: Apress, 2003, xviii, 364 p.

LAUDON, Kenneth C.; LAUDON, Jane Price. **Sistemas de informação**. Tradução Dalton Conde de Alencar. Rio de Janeiro: LTC, 1999.

MENEZES, R. **Câmara Brasileira de Comércio Eletrônico**. São Paulo, [2003?]. Disponível em: <http://www.camara-e.net/newsletter/2004/newsletter01setembro04.htm>. Acesso em: 12 set. 2005.

MONTENEGRO, C.; PEREIRA, C. **Java de ponta a ponta do J2ME ao J2EE.** Mundo Java,Rio de Janeiro, n. 12, p. 28-43, 2005.

PAMPLONA, V. F. **Um protótipo de motor de jogos 3D para dispositivos móveis com suporte a especificação mobile 3D graphics API for J2ME**. 2005. 83 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

ROSA, F. **Protótipo de um diário de classe em dispositivos móveis utilizando J2ME**. 2005. 99 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

SOUZA, Ricardo Araújo de; LIMA-CARDOSO, André. **UML aplicada da teoria a implementação**. Rio de Janeiro: Editora Ciência Moderna Ltda., 2007.

SCHMITT JUNIOR, Arno José. **Protótipo de front end de controle de acesso usando J2ME.** 2004. 69 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) -Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

TEIXEIRA, C. A. **Dois bilhões de celulares no mundo**. [S.l], 2005. Disponível em: <http://www.torque.com.br/index.php?modulo=textos&secao=artigos&codTexto=1653&pagina=1&sequencia=1&codCategoria=#>. Acesso em: 23 out. 2010.

TOPLEY, Kim. **J2ME in a nutshell :** a desktop quick reference**.** Cambridge: O`Reilly, c2002. xv, 450 p.

WHITE, James, HEMPHILL, David. **Java 2 Micro Edition**: Java in small things. Greenwich:Manning, 2002, 479 p.

YOSHIMURA, B. H. **O que é smartphone?**. [S.l.], 2006. Disponível em: http://www.linkgratis.com.br/materia/O_Que_e_Smartphone/ Acesso em: 27 out. 2010.

APÊNDICE A – Detalhamento dos casos de uso

Nos quadros a seguir estão as descrições dos casos de uso do sistema.

No Quadro 8 apresenta-se o caso de uso "Efetuar Login".

Nome do Caso de Uso	Efetuar Login
Descrição	Cliente acessa aplicação via celular e informa dados para login e senha armazenados no cadastro do cliente da aplicação servidor.
Ator	Cliente
Pré-condição	Cliente deve estar cadastrado no banco de dados da aplicação servidor.
Fluxo principal	a) Cliente preenche seu login e sua senha; b) Sistema valida os dados de login e senha do usuário; c) Sistema lista os veículos em locação.
Fluxo alternativo (a)	• nome de usuário e/ou senha inválido(s) • alerta com mensagem “usuário ou senha incorretos” é mostrada.
Pós-condição	Cliente entra conectado ao sistema.

Quadro 8 – Descrição do caso de uso Login

No Quadro 9 apresenta-se o caso de uso "Gera solicitação Reserva Veiculo".

Nome do Caso de Uso	Gera solicitação Reserva Veiculo
Descrição	Cliente seleciona o veiculo e datas para reserva. Serão mantidos os dados: Id da reserva, ClienteId, VeiculoId, DataRetirada, hora da retirada, DataDevolução, hora da devolução.
Ator	Cliente
Pré-condição	Cliente deve fazer *login* no sistema.
Fluxo principal	O sistema informa os veículos disponíveis;
Cenário – Visualização	Sistema mostra os registros de veículos disponíveis para reserva;
Cenário – Edição	1. Sistema mostra registros da reserva;
Cenário – Inclusão	1. Sistema mostra registros reservados; 2. Sistema libera opção para nova reserva
Pós-condição	Cliente visualizou / Incluiu uma reserva.

Quadro 9 – Descrição do caso de uso Gera solicitação Reserva Veiculo

APÊNDICE B – Utilização da biblioteca kSOAP2 e validação de login

Implementação de um método da classe AcessarWS que utiliza a biblioteca kSOAP2.

```
public boolean autenticarUsuario(String user, String pwd){
boolean ret = false;
//Cria um objeto SOAP (XML) usando a url e o nome do método
a chamar.
SoapObject client = new SoapObject(url,
"autenticarUsuario");

/*Adiciona propriedades (parâmetros) conforme o método
aguarda
* É muito importante aqui acompanhar os tipos dos parâmetros
* esperados. Neste caso os parâmetros são Strings, mas há
métodos
* que o parâmetro precisa ser Integer e isso é muito
importante
* para o sucesso da operação.*/

client.addProperty("user", user);
client.addProperty("pwd", pwd);

//Cria um objeto Http para transportar via web o objeto
SOAP.
HttpTransport ht = new HttpTransport(url,
"autenticarUsuario");

//Como esse método retorna um Integer (objeto, não um
'int'), é
//importante criar esse objeto de retorno .

Integer clienteID;
try{
//Faz o call usando o objeto SOAP e o transportador Http
clienteID = (Integer)(ht.call(client));

//O ws foi construído para devolver -1 se o cliente não for
validado.
if(clienteID.intValue()>-1){
//Se a validação ocorreu o ID do cliente retornou e será
guardado
//na classe de controle para uso do sistema.
main.setClienteID(clienteID.intValue());
ret = true;
}
}
```

Quadro 10 – Exemplo de utilização da biblioteca kSOAP2

Implementação de um método da classe Login que valida Usuario e Senha na aplicação Cliente.

```
public void commandAction(Command command, Displayable
displayable) {
if (command == fecharCmd) {
main.finalizar();
}
else if (command == enviarCmd) {
if((!USER.getString().equalsIgnoreCase(""))
&& (!PSWD.getString().equalsIgnoreCase(""))){

if              (main.ws.autenticarUsuario(USER.getString(),
PSWD.getString())) {
main.setUsuario(USER.getString());
main.setSenha(PSWD.getString());

main.listarCarrosDisponiveis();
} else {
Command okCommand = new Command("Ok", Command.OK, 0);
Alert alert = new Alert("Erro", "Usuário ou senha
incorretos.", null, AlertType.ERROR);
alert.addCommand(okCommand);
main.getDisplay().setCurrent(alert);
}
}
else
{
```

Quadro 11 – Exemplo de código fonte da validação de login