

**LIGA DE ENSINO DO RIO GRANDE DO NORTE
FACULDADE NATALENSE PARA O DESENVOLVIMENTO DO RIO GRANDE DO NORTE**

**PROJETO DE APLICAÇÃO PARA DISPOSITIVOS MÓVEIS
UTILIZANDO WEB SERVICES E J2ME**

George Carlos Lucena de Oliveira

**PROJETO DE APLICAÇÃO PARA DISPOSITIVOS MÓVEIS
UTILIZANDO WEB SERVICES E J2ME**

Bacharelado em Sistema de Informação da
Faculdade Natalense para o Desenvolvimento
do Rio Grande do Norte como requisito
obrigatório para a obtenção do grau de
Bacharel em Sistema de Informação

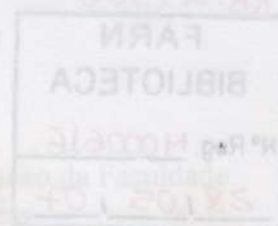
Orientador:

Felipe Araújo Alcino, M.Sc

**NATAL – RN
2006**

GEORGE CARLOS LUCENA DE OLIVEIRA
GEORGE CARLOS LUCENA DE OLIVEIRA

PROJETO DE APLICAÇÃO PARA DISPOSITIVOS MÓVEIS
UTILIZANDO WEB SERVICES E J2ME
**PROJETO DE APLICAÇÃO PARA DISPOSITIVOS MÓVEIS
UTILIZANDO WEB SERVICES E J2ME**



Monografia apresentada ao curso de Bacharelado em Sistema de Informação da Faculdade Natalense para o Desenvolvimento do Rio Grande do Norte como requisito obrigatório para a obtenção do grau de Bacharel em Sistemas de Informação.

Orientador:

Fellipe Araújo Aleixo, M.Sc.

GEORGE CARLOS LUCENA DE OLIVEIRA

PROJETO DE APLICAÇÃO PARA DISPOSITIVOS MÓVEIS
UTILIZANDO WEB SERVICES E J2ME

RB. 29306

FARN
BIBLIOTECA
Nº Reg. <u>M000616</u>
<u>28/05/07</u>

004.738.5

045p

2006

Ex. 01

Catálogo na Publicação – Biblioteca da FARN
Setor de Processos Técnicos

Oliveira, George Carlos Lucena de

Projeto de aplicação para dispositivos móveis utilizando Web Service e J2ME /
George Carlos Lucena de Oliveira. – Natal: FARN, 2006.
53f.

Orientador: Felipe Araújo Aleixo (M.Sc.)

Monografia (Bacharelado em Sistemas de Informação) – Faculdade Natalense
para o Desenvolvimento do Rio Grande do Norte.

1. Internet em Telefonia Móvel – Monografia. 2. Linguagem de Programação –
Monografia. 3. Sistema de Consulta a Automóveis – Monografia. I. Título.

RN/FARN/BC

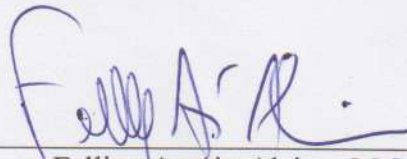
CDU 004.738.5

NATAL - RN
2006

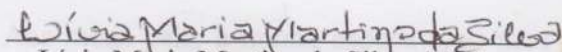
GEORGE CARLOS LUCENA DE OLIVEIRA

**PROJETO DE APLICAÇÃO PARA DISPOSITIVOS MÓVEIS
UTILIZANDO WEB SERVICES E J2ME**

Monografia apresentada ao Curso de Bacharelado em Sistemas de Informação da Faculdade
Natalense para o Desenvolvimento do Rio Grande do Norte, aprovada em _____ de
_____ de _____, pela banca examinadora composta pelos
seguintes membros.



Felipe Araújo Aleixo, M.Sc.
Orientador



Livia Maria Martins da Silva, M.Sc.

Membro examinador



Antônio de Pádua de Miranda Henriques

Membro examinador

AGRADECIMENTOS

Primeiramente a Deus por ter me ajudado a encontrar a profissão certa. Aos meus pais, pelo incentivo e por terem sempre acreditado no meu ideal. A minha esposa pelo carinho, do qual sou de dar ao nosso filho, pela minha ausência. Em especial ao meu amado filho, George Victor, que mesmo sem entender, devido a pouca idade, soube ter paciência e compreensão nos dias em que precisei ficar ausente para estudar. Ao professor Gleydon Lima por auxiliar na escolha desta monografia. Ao meu orientador Felipe Almeida que sempre se mostrou disponível nas horas que precisei de uma orientação e auxílio para conclusão desta monografia.

Aos meus pais, a minha esposa e ao meu amado filho dedico este trabalho, como resultado de meu esforço e também do apoio que deles recebi em todos os projetos de minha vida, sempre me incentivando, confiando no meu potencial e fazendo o possível e impossível por mim.

AGRADECIMENTOS

Primeiramente a Deus por ter me ajudado a encontrar a profissão certa. Aos meus pais, pelo incentivo e por terem sempre acreditado no meu ideal. A minha esposa pela dedicação que teve de dar ao nosso filho, pela minha ausência. Em especial ao meu amado filho, George Victor, que mesmo sem entender, devido a pouca idade, soube ter paciência e compreensão nos dias em que precisei ficar ausente para estudar. Ao professor Gleydson Lima por auxiliarme na escolha desta monografia. Ao meu orientador Fellipe Aleixo que sempre se mostrou disponível nas horas que precisei de uma orientação e auxílio para conclusão desta monografia.

Palavras-chave: Dispositivos Móveis, Web Services, J2ME.

RESUMO

A Internet nos dias atuais proporciona cada vez mais uma expansão nos serviços prestados ao público, em geral, conectado a ela, entre esses serviços está o serviço de consultas a veículos, disponibilizado no site do DETRAN. Tendo em vista que para o usuário venha a dispor deste serviço, ele precisará ter no mínimo um computador e que este, tenha acesso à internet e que esta não é um serviço tão barato e tão conveniente, surgiu a idéia desta monografia que é apresentar uma outra solução. Tendo em vista a crescente expansão de aplicações para dispositivo móvel, especificamente celular que é uma alternativa de menor custo, maior portabilidade, maior praticidade e agilidade, pois basta o usuário fazer uma conexão com a internet para poder navegar. Pensando nisso, surgiu a idéia de propor um projeto de uma aplicação para realizar uma consulta se um determinado veículo tem algum tipo de impedimento, infrações ou "Nada Consta", que é o objetivo desta monografia, e, que tem como título "Projeto de Aplicação para Dispositivos Móveis Utilizando Web Services e J2ME". Ao final, conclui-se que é viável a utilização do celular para realizar as consultas de "Nada Consta", Infrações ou Impedimentos, dos veículos, melhorando ainda mais os serviços prestados por àquele órgão aos seus usuários e oferecendo como vantagem uma forma mais prática e ágil a essas informações.

Palavras-chave: Dispositivos Móveis. Web Services. J2ME.

ABSTRACT

The Internet in the current days provides each time plus an expansion in the given services to the public, in general, connected it, between these services is the service of consultations the vehicles, in the site of the DETRAN. In view of that to the user he comes to make use of this service, it will need to have a computer at least and that this, has access to the Internet and that this is not a service so cheap and so convenient, appeared the idea of this monograph that is to present one another solution. In view of the increasing expansion of applications for mobile device, specifically cellular that is an alternative of lesser cost, greater portability, greater practicable and agility, therefore the user is enough to make a connection with the Internet to be able to sail. Thinking about this, he appeared to the idea to consider a project of an application to carry through a consultation if one definitive vehicle has some type of impediment, infractions or "Nothing It consists", that it is the objective of this monograph, and, that has as heading "Project of Application for Mobile Devices Using Web Services and J2ME". To the end, I was concluded that the use of the cellular one is viable to carry through the consultations of "Nothing Consists", Infractions or Impediments, of the vehicles, improving still more the services given for that agency to its users and offering as advantage a more practical and agile form to these information.

Figura 17 - Estrutura da Aplicação Exemplo	40
Figura 18 - Código da Aplicação Alô Detran	43
Figura 19 - Arquitetura Proposta para o Sistema	44
Key Words: Devices Mobile; Web Services; J2ME.	45
Figura 20 - Caso de Uso Alô Nível	46
Figura 22 - Diagrama de Classes	49

LISTA DE FIGURAS

Figura 1 - Exemplo de Diagrama de Casos de Uso	15
Figura 2 - Exemplo de Diagrama de Classes.....	17
Figura 3 - Exemplo de Diagrama de Colaboração.....	17
Figura 4 - Exemplo do Diagrama de Sequência	19
Figura 5 - Arquitetura de um <i>Web Service</i> . Fonte: Hendricks (2002, p.10).....	21
Figura 6 - Sobreposição de Camadas de Comunicação de um <i>Web Service</i>	22
Figura 7 - Exemplo da Estrutura de Uma Mensagem SOAP	23
Figura 8 - Exemplo dos Principais Elementos de um WSDL	25
Figura 9 - Exemplo de Código WSDL.....	26
Figura 10 - Exemplo de Código em Java	30
Figura 11 - Exemplo de Código J2ME	33
Figura 12 - Exemplo da Estrutura do JDBC.....	34
Figura 13 - Aplicação Exemplo - Tela Novo Projeto	36
Figura 14 - Aplicação Exemplo - Tela Nome do Projeto.....	37
Figura 15 - Aplicativo Exemplo - Tela Emulador da Plataforma.....	38
Figura 16 - Aplicativo Exemplo - Tela Modelo dos Celulares.....	39
Figura 17 - Estrutura da Aplicação Exemplo	40
Figura 18 - Código da Aplicação Alô Detran.....	42
Figura 19 - Arquitetura Proposta para o Sistema.....	44
Figura 20 - Diagrama de Caso de Uso do Sistema	45
Figura 21 - Caso de Uso Alto Nível.....	46
Figura 22 - Diagrama de Classes	49

JDBC - Java Data Base Connectivity

VM - Virtual Machine

SQL - Structure Query Language

ODBC - Open Data Base Connectivity

IDE - Integrated Development Environment

PDA - Personal Digital Assistant

LISTA DE SIGLAS

J2ME - Java Micro Edition	11
UML - Linguagem de Modelagem Unificada	11
JDBC - Java Database Connectivity	12
API - Application Programming Interface	12
XML - Extensible Markup Language	13
SOAP - Simple Object Access Protocol	14
HTTP - Hypertext Transfer Protocol	14
RPC - Remote Procedure Call	15
UDDI - Universal Description Discovery, and Integration	15
SMTP - Simple Mail Transfer Protocol	18
FTP - File Transfer Protocol	20
SOA - Service-Oriented Architectures	22
WSDL - Web Service Description Language	24
JVM - Java Virtual Machine	26
API - Application Programming Interface	27
JDBC - Java Data Base Connectivity	28
VM - Virtual Machine	30
SQL - Structure Query Language	33
ODBC - Open Data Base Connectivity	35
IDE - Integrated Development Environment	36
PDA - Personal Digital Assistant	36
4.2.3 Arquitetura do Sistema	44
4.2.4 Diagramas de Caso de Uso	45
4.2.4.1 Caso de Uso de Alto Nível "ConsultarPlaca"	45
4.2.4.2 Caso de Uso Expandido "ConsultarPlaca"	45
4.2.5 Diagrama de Classes Proposto	48
5. CONCLUSÃO	50
REFERÊNCIAS	51

SUMÁRIO

1 INTRODUÇÃO	11
1.1 PROBLEMA DE PESQUISA	11
1.2 OBJETIVOS	12
1.2.1 Objetivo Geral	12
1.2.2 Objetivos Específicos	12
1.3. JUSTIFICATIVA	13
2 REFERENCIAL TEÓRICO	14
2.1 UML	14
2.1.1 Diagrama de Casos de Uso	15
2.1.2 Diagrama de Classes	16
2.1.3 Diagrama de Colaboração	17
2.1.4 Diagrama de Sequência	18
2.2 XML	20
2.3 WEB SERVICE	20
2.3.1 SOAP	22
2.3.2 WSDL	24
2.3.3 UDDI	26
2.4 JAVA	27
2.4.1 Principais Características do Java	28
2.5 J2ME	30
2.6 JDBC	33
3 METODOLOGIA	35
4 RESULTADO OBTIDO	36
4.1 DESENVOLVIMENTO DE APLICAÇÕES COM J2ME	36
4.2 ANÁLISE E PROJETO DO SISTEMA DE CONSULTA A PLACAS DE AUTOMÓVEIS APLICANDO AS TECNOLOGIAS DE WEBSERVICE E J2ME	43
4.2.1 Requisitos Funcionais	43
4.2.2 Requisitos não Funcionais	43
4.2.3 Arquitetura do Sistema	44
4.2.4 Diagramas de Caso de Uso	45
4.2.4.1 Caso de Uso de Alto Nível "ConsultarPlaca"	45
4.2.4.2 Caso de Uso Expandido "ConsultarPlaca"	45
4.2.5 Diagrama de Classes Proposto	48
5. CONCLUSÃO	50
REFERÊNCIAS	51

1 INTRODUÇÃO

A computação se aproxima cada vez mais de seus usuários, tornando-se mais presente na vida das pessoas. A evolução dos dispositivos portáteis é constante, e permite serviços que alteram as relações de trabalho e sociais de funcionários em corporações, e também as próprias relações de usuários finais com a computação (RIBEIRO, 2004, p. 14).

A crescente expansão de aplicações para dispositivos móveis utilizando a Internet tem se tornado um grande atrativo para as empresas modernizarem suas ações. Essas tecnologias levam as empresas a serem mais competitivas, visando atender aos princípios básicos da administração, como redução de custo, agilidade e satisfação dos seus clientes.

Pensando nisso, esta monografia propõe um projeto de uma aplicação, tendo como exemplo um sistema de consulta a placas de automóveis, para funcionar em quaisquer dispositivos móveis, desde que tenham acesso à Internet, utilizando para isso as tecnologias de *Web Services* (Serviços Web) e *J2ME* (*Java Micro Edition*).

O sistema proposto utiliza a informação de uma placa de veículo, ou qualquer outro código que a substitua, para fazer uma simulação de consulta à base de dados do DETRAN; órgão responsável pelas informações dos veículos, tais como verificação de multas, impedimentos ou “nada consta”, tornando o órgão mais ágil na prestação dos serviços prestados aos seus clientes.

1.1 PROBLEMA DE PESQUISA

Segundo um estudo realizado pela primeira edição dos Indicadores VIVO do Mercado Brasileiro de Telefonia Móvel, revela que o mercado brasileiro deve atingir 101 milhões de aparelhos vendidos no ano de 2006 e um índice de penetração de 54%. Em 1998 foram registrados 7 milhões de aparelhos. Quanto à faixa etária, o estudo indica uma grande concentração de usuários com idade entre 14 e 30 anos (50%), enquanto 38% têm entre 31 e 50 anos e 12% estão na faixa de mais de 51 anos, onde, nesta última faixa etária teve o maior crescimento entre junho de 2005 e junho de 2006 (53%), (BRAUN, 2006).

Atualmente, as empresas e/ou órgãos têm investido cada vez mais em tecnologias de software para atender ao seu crescimento bem como para solucionar os problemas decorrentes desta evolução.

Dai surge o problema de estudo, que é o foco desta monografia, será que não teria uma forma de agilizar, ainda mais, o acesso às informações dos veículos, no tocante as multas, impedimentos e “nada consta”?

Pode ser utilizado um telefone celular, que é um dispositivo móvel, para que os usuários tenham uma maior flexibilidade e agilidade na obtenção de informações a respeito dos veículos?

1.2 OBJETIVOS

1.2.1 Objetivo Geral

A presente monografia tem por objetivo um projeto de aplicação que utiliza dispositivo móvel para pesquisar se um determinado veículo possui algum tipo de impedimento, multa ou “nada consta”, informações estas que são acessadas de uma base de dados, via Web Service.

1.2.2 Objetivos Específicos

Para atingir o objetivo central do trabalho, foi necessário satisfazer os seguintes objetivos específicos:

1. Realizar estudo bibliográfico sobre UML (Linguagem de Modelagem Unificada); mais especificamente os artefatos a serem utilizados no projeto do sistema proposto;
2. Estudos bibliográficos da plataforma J2ME, que será a plataforma utilizada para o desenvolvimento da aplicação;
3. Estudo bibliográfico da tecnologia de Web Services, este estudo se faz necessário para dar ao projeto uma interoperabilidade entre as distintas plataformas existentes;
4. Estudo bibliográfico sobre JDBC que é a interface de acesso ao banco de dados;
5. Realizar a análise do sistema proposto.

1.3. JUSTIFICATIVA

Segundo a Pesquisa Nacional por Amostra de Domicílios (Pnad) 2005, divulgada no dia 15 de novembro, pelo Instituto Brasileiro de Geografia e Estatística (IBGE), o número de vendas de celulares no Brasil está crescendo cada vez mais, por isso, este projeto tenta dar mais uma alternativa para auxiliar pessoas a poderem fazer pesquisas sobre os veículos, através deste tipo de dispositivo móvel.

Com o uso do sistema o usuário terá maior agilidade na consulta dos veículos, pois não precisará mais ter acesso a um computador ou, até mesmo, ter que ir ao órgão em questão para buscar tais informações.

Além das facilidades já apresentadas, o aplicativo terá ainda algumas vantagens, entre as quais podem ser destacadas:

- Uso do telefone celular, permitindo seu uso em qualquer lugar, desde que tenha acesso à rede de telefonia celular e que seu aparelho tenha os requisitos necessários para poder fazer a conexão;
- A tecnologia utilizada oferece agilidade, flexibilidade, portabilidade aos usuários;
- Redução de custos para se obter tais informações, uma vez que passa a ser dispensado o uso de um computador que tenha acesso à Internet.

Tendo uma visão mais acadêmica, este projeto implica no emprego dos conhecimentos adquiridos ao longo do curso, além de representar, diante das pesquisas realizadas, o melhor uso da tecnologia hoje existente como solução ao problema proposto.

- A UML, na implementação de um sistema pode fazer uso de nove tipos de diagramas diferentes:
- Diagrama de casos de uso
 - Diagrama de classes
 - Diagrama de objetos
 - Diagrama de estado
 - Diagrama de sequência
 - Diagrama de colaboração
 - Diagrama de atividade
 - Diagrama de componente
 - Diagrama de execução

2 REFERENCIAL TEÓRICO

2.1 UML

No desenvolvimento de qualquer aplicativo onde os desenvolvedores e analistas de sistemas possam visualizar os produtos dos seus trabalhos em diagramas padronizados, junto com uma notação gráfica, se faz uso da UML.

Segundo Larman (2004, p.34), "UML é uma linguagem para especificar, visualizar, construir e documentar os artefatos de sistemas de software, bem como para modelar negócios e outros sistemas que não sejam de software".

De acordo com Laureano (apud BOOCH, RUMBAUGH e JACOBSON, 2000, p.19), define que:

UML é uma linguagem gráfica para a visualização, especificação, construção e documentação de artefatos de sistemas complexos de software. UML proporciona uma forma padrão para a preparação de planos de arquitetura de projetos de sistemas, incluindo aspectos conceituais tais como processos de negócios e funções do sistema, além de itens concretos como as classes escritas em determinada linguagem de programação, esquemas de bancos de dados e componentes de software reutilizáveis.

A UML servirá para demonstrar graficamente como o sistema proposto irá funcionar, mostrando como o usuário irá interagir com o mesmo.

A UML, na implementação de um sistema pode fazer uso de nove tipos de diagramas diferentes:

- Diagrama de casos de uso
- Diagrama de classes
- Diagrama de objetos
- Diagrama de estado
- Diagrama de sequência
- Diagrama de colaboração
- Diagrama de atividade
- Diagrama de componente
- Diagrama de execução

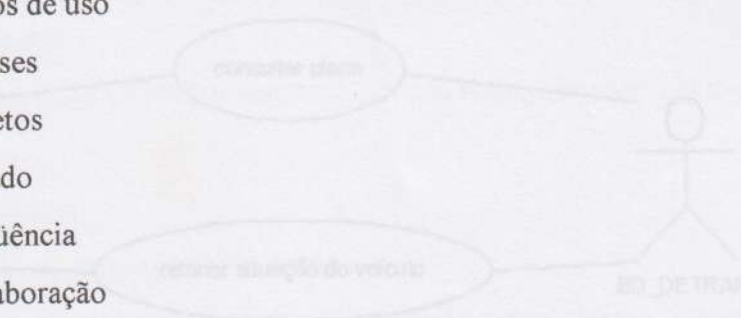


Figura 1 - Exemplo de Diagrama de Casos de Uso
Fonte: O próprio autor

Na figura 1 é apresentado um exemplo de diagrama de casos de uso. Neste exemplo, é mostrado o ator usuário representando qualquer ser humano que interage com o sistema.

Para elaboração deste projeto, serão utilizados apenas quatro diagramas para ilustrar o projeto do sistema proposto, que são:

- Diagrama de casos de uso;
- Modelo conceitual – diagrama de classes;
- Diagrama de colaboração;
- Diagrama de sequência.

2.1.1 Diagrama de Casos de Uso

A modelagem de um diagrama de caso de uso é uma técnica usada para descrever e definir os requisitos funcionais de um sistema.

É um diagrama usado para se identificar como o sistema se comporta em várias situações que podem ocorrer durante sua operação. Descrevem o sistema, seu ambiente e a relação entre os dois. Os componentes deste diagrama são os atores e os casos de uso.

Os atores representam o papel de uma entidade externa ao sistema, os usuários ou outros sistemas externos que façam interação.

Um caso de uso é uma sequência de ações que o sistema executa e produz um resultado de valor para o ator, representando o diálogo entre atores e o sistema. Sempre um caso de uso é iniciado por um ator para invocar certa funcionalidade do sistema.

Segundo Booch, Rumbaugh e Jacobson (2000, p.220), um caso de uso “é uma descrição de um conjunto de seqüências de ações, inclusive variantes, que um sistema executa para produzir um resultado de valor observável por um ator”.



Figura 1 - Exemplo de Diagrama de Casos de Uso
Fonte: O próprio autor

Na figura 1 é apresentado um exemplo de diagrama de casos de uso. Neste exemplo, é ilustrado o ator usuário representando qualquer ser humano que interage com o sistema.

Também é ilustrado o ator BD_DETRAN representando um outro sistema com o qual o sistema exemplo interage – o sistema de gerenciamento de banco de dados do DETRAN. Neste mesmo exemplo, o sistema ilustrado conta com duas funcionalidades principais: “consultar placa” e “retornar situação do veículo”.

Segundo Larman (2002, p.69), “Um ator é algo com comportamento, tal como uma pessoa (que é identificada por seu papel), um sistema de computador ou uma organização”.

2.1.2 Diagrama de Classes

Um diagrama de classes é um diagrama que mostra um conjunto de classes, interfaces e colaborações e seus relacionamentos. Graficamente, um diagrama de classes é uma coleção de vértices e arcos (BOOCH, 2000, p.105).

De acordo com Booch; Rumbaugh; Jacobson, (2000, p.94), Os diagramas de classes ilustram a visão estática do projeto de um sistema.

Para se criar um diagrama de classes deverá seguir os seguintes passos:

- Identificar todas as classes participantes da solução em software, que deverá ser desenhada em um diagrama;
- Duplicar os atributos dos conceitos associados do modelo conceitual;
- Acrescentar nomes aos métodos aos através da análise dos diagramas de interação (seqüência e colaboração);
- Acrescentar informações sobre os tipos de atributos e métodos;
- Acrescentar as associações para suportar as visibilidades de atributo requeridas;
- Acrescentar linhas de relacionamentos de dependência para indicar visibilidade que não sejam de atributos.

A notação UML para Classes de Objetos é denotada por um retângulo com três divisões:

- Nome da Classe;
- Atributos relacionados;
- Operações, que são as mensagens ou métodos.

A figura 2, abaixo, mostra um exemplo contendo cinco classes relacionadas, com seus atributos e métodos.

Figura 3 - Exemplo de Diagrama de Colaboração
Fonte: O próprio autor

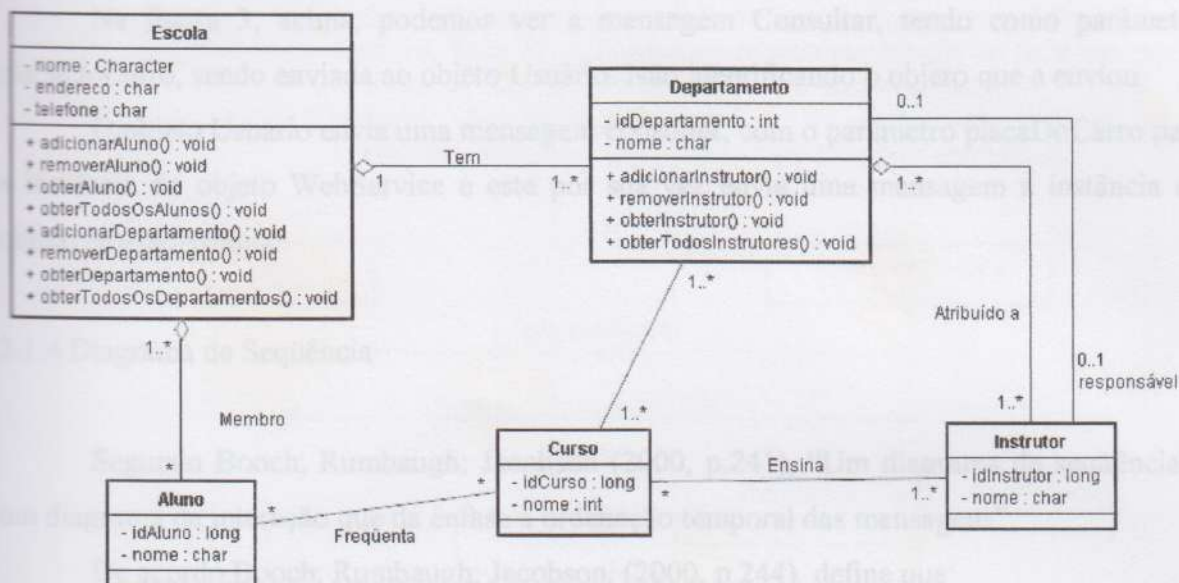


Figura 2 - Exemplo de Diagrama de Classes
Fonte: BOOCH (2000)

2.1.3 Diagrama de Colaboração

Um diagrama de colaboração mostra os objetos e seus relacionamentos e a troca de mensagens entre eles.

Segundo Booch; Rumbaugh; Jacobson (2000, p.366), “Uma colaboração é uma sociedade de classes, interfaces e outros elementos que trabalham em conjunto para fornecer algum comportamento cooperativo maior do que a soma de todas as suas partes”.

De acordo com Matos (2002, p.62), “O diagrama de colaboração fixa a atenção em como os objetos estão se organizando para efetuar uma tarefa. É como se uma cadeia de ajuda fosse efetivada com a intenção de que um desejo fosse alcançado”.

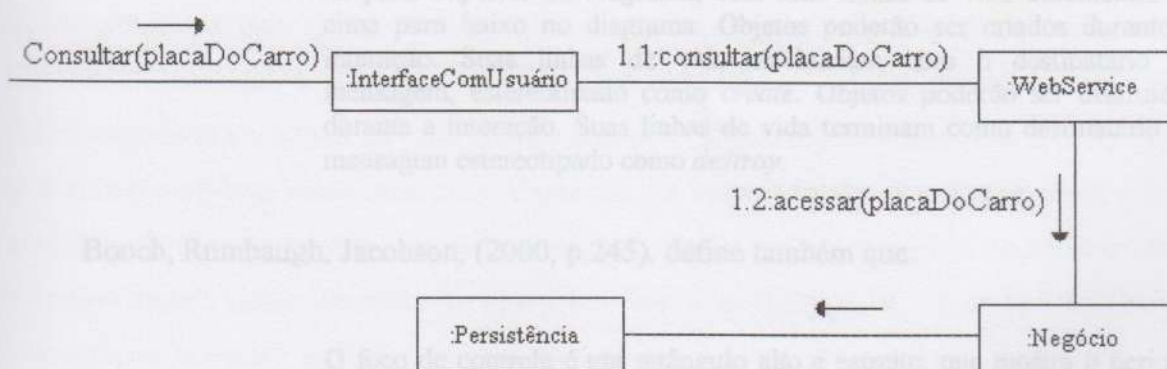


Figura 3 - Exemplo de Diagrama de Colaboração
Fonte: O próprio usuário

Na figura 3, acima, podemos ver a mensagem Consultar, tendo como parâmetro placaDoCarro, sendo enviada ao objeto Usuário. Não identificando o objeto que a enviou.

O objeto Usuário envia uma mensagem consultar, com o parâmetro placaDoCarro para a instância do objeto WebService e este por sua vez envia uma mensagem a instância do objeto AcessoABanco.

2.1.4 Diagrama de Seqüência

Segundo Booch; Rumbaugh; Jacobson (2000, p.241), “Um diagrama de seqüência é um diagrama de interação que dá ênfase à ordenação temporal das mensagens”.

De acordo Booch; Rumbaugh; Jacobson, (2000, p.244), define que:

Um diagrama de seqüências é formado colocando-se primeiro os objetos que participam da interação no nível superior do diagrama, ao longo do eixo X. Tipicamente, objeto que inicia a interação é colocado à esquerda e objetos mais subordinados vão crescendo à direita. A seguir, as mensagens que esses objetos enviam e recebem são colocadas ao longo do eixo Y, em ordem crescente de tempo, de cima para baixo. É isso que proporciona uma clara indicação visual do fluxo de controle ao longo do tempo.

Os diagramas de seqüências são identificados por duas características básicas que é a linha de vida do objeto e o foco de controle.

Booch; Rumbaugh; Jacobson, (2000, p.244), define que:

Figura 3- Exemplo de Diagrama de Seqüência

A linha de vida do objeto é a linha tracejada vertical que representa a existência de um objeto em um período de tempo. Esses objetos terão existência igual à duração da interação; assim, esses objetos são alinhados na parte superior do diagrama, com suas linhas de vida desenhadas de cima para baixo no diagrama. Objetos poderão ser criados durante a interação. Suas linhas de vida se iniciam com o destinatário da mensagem, estereotipado como *create*. Objetos poderão ser destruídos durante a interação. Suas linhas de vida terminam como destinatário da mensagem estereotipado como *destroy*.

Booch; Rumbaugh; Jacobson, (2000, p.245), define também que:

O foco de controle é um retângulo alto e estreito, que mostra o período durante o qual um objeto está desempenhando uma ação, diretamente ou por meio de um procedimento subordinado. Aparte superior do retângulo é alinhado com o início da ação; a parte inferior é alinhada com sua conclusão, podendo ser marcada com uma mensagem de retorno. Pode-se

exibir o alinhamento de um foco de controle ligeiramente à direita de seu foco pai. Para ser especialmente preciso em relação ao local em que se encontra o foco de controle, pode-se também sombrear a região do retângulo durante a qual o método do objeto está realmente sendo computado, sem que o objeto tenha sido passado a outro objeto.

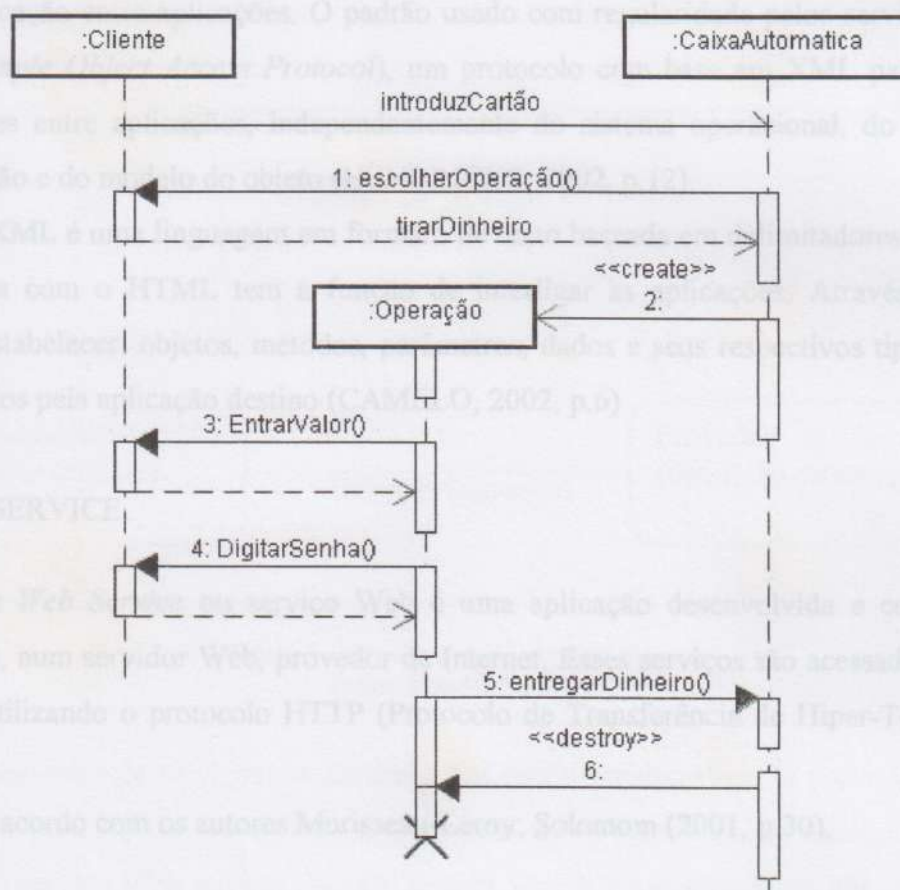


Figura 4 - Exemplo do Diagrama de Seqüência

Fonte: Anquetil (2002)

O exemplo de diagrama de seqüência, mostrado na figura 4 acima, temos um cliente fazendo um saque num caixa eletrônico. Os objetos, neste exemplo, são identificados como “Cliente”, “Operação” e “CaixaAutomatica”, representados pela caixa retângulo. As linhas verticais representam a vida de cada objeto, quando o objeto recebe o controle, a linha muda seu formato para uma caixa retangular e quando ela volta a perder o controle, volta a ficar tracejada. A figura mostra várias mensagens, como envio de sinal podemos citar “introduzCartão”, como chamada de operações “escolherOperação()”, “entregarDinheiro()”, como retorno de valor “Senha”, além de uma mensagem de criação e outra de destruição.

E, segundo os autores Todd e Szolkowski (2003), “Web Service é um componente de software que pode ser invocado por meio de uma rede HTTP, sendo a invocação feita por

2.2 XML

Como podemos observar, na figura 04, o XML (*Extensible Markup Language*) é a base da comunicação das camadas. Essa é a camada que define o formato de mensagem usado na comunicação entre aplicações. O padrão usado com regularidade pelos serviços Web é o SOAP (*Simple Object Access Protocol*), um protocolo com base em XML para a troca de informações entre aplicações, independentemente do sistema operacional, do ambiente de programação e do modelo do objeto (HENDRICKS, 2002, p.12).

O XML é uma linguagem em formato de texto baseada em delimitadores, por isso sua semelhança com o HTML tem a função de interligar as aplicações. Através do XML é possível estabelecer: objetos, métodos, parâmetros, dados e seus respectivos tipos que serão interpretados pela aplicação destino (CAMELO, 2002, p.6)

2.3 WEB SERVICE

Um *Web Service* ou serviço Web é uma aplicação desenvolvida e colocada, para publicação, num servidor Web, provedor de Internet. Esses serviços são acessados através da Internet, utilizando o protocolo HTTP (Protocolo de Transferência de Hiper-Texto) para se comunicar.

De acordo com os autores Morisseau-Leroy; Solomom (2001, p.30),

Servidor Web é um programa que recebem pedidos que cheguem através de uma rede, executa alguma lógica baseada nos parâmetros do pedido e retorna os resultados para o aplicativo cliente. Normalmente, esses pedidos são enviados por um ser humano, que clica nos hyperlinks de página da Web apresentada por um navegador da Web. A resposta gerada pelo servidor da Web é manipulada e apresentada pelo navegador.

Segundo Souza (2005, p.5), "*Web Services* são componentes de software que independem de implantação ou de plataforma e podem ser descritos, publicados e invocados sobre uma rede através de mensagens padrão XML".

Já Paulan (2004) descreve, "*Web Service* pode ser considerado como um serviço disponível na WEB, porém sem telas gráficas, que utilizam chamadas e retornos em formato XML, utilizando padrão SOAP".

E, segundo os autores Todd e Szolkowski (2003), "*Web Service* é um componente de software que pode ser invocado por meio de uma rede HTTP, sendo a invocação feita por

meio da tecnologia XML, e a plataforma e a linguagem do componente de software devem ser transparentes para o invocador do componente” (HENDRICKS, 2002, p. 10).

De acordo com Hendricks (2002, p.10) identifica três tipos de papéis em um ambiente de serviços Web típico, bem como as operações que eles executam para fazer os serviços funcionarem. Ver diagrama abaixo:

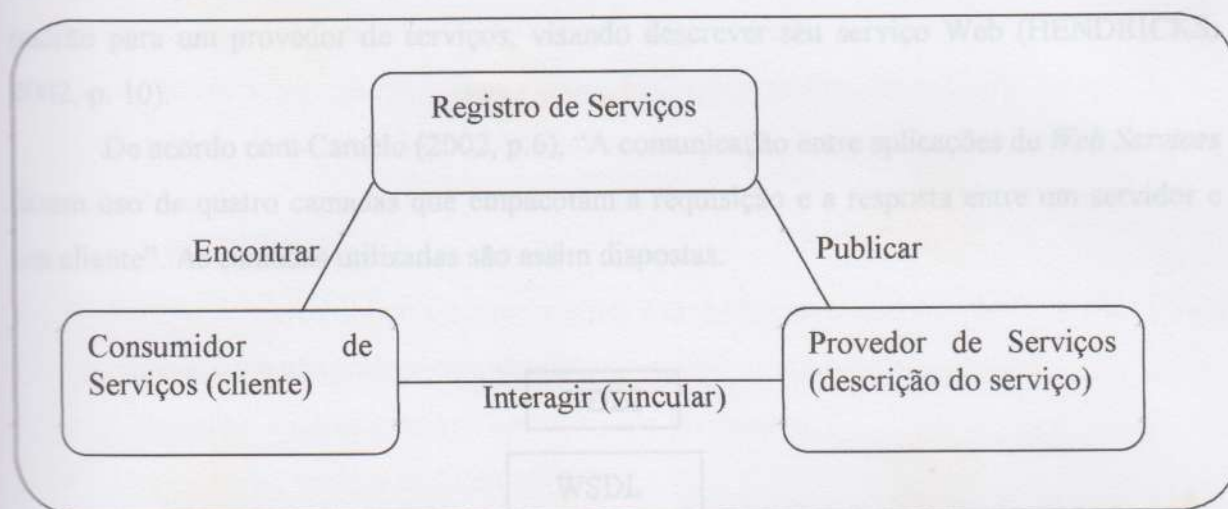


Figura 5 - Arquitetura de um Web Service.
Fonte: Hendricks (2002)

O provedor de serviços é a entidade que cria o serviço Web. O provedor precisa ter duas coisas para ter acesso a todo o potencial de serviço Web. Primeiramente, precisa descrever o serviço Web em um formato padrão, que seja compreensível por qualquer outro sistema que possa usar esse serviço Web, em segundo, para alcançar um grande público, ele deve publicar os detalhes sobre seu serviço Web em um registro central que esteja publicamente disponível para todos os interessados (HENDRICKS, 2002, p. 10).

O consumidor de serviços pode ser qualquer empresa criada por um provedor de serviços e que utilize um serviço Web. O consumidor de serviços é capaz de conhecer um serviço Web a partir da descrição disponibilizada pelo provedor de serviços. Para recuperar os detalhes, o consumidor de serviços realiza uma pesquisa sobre o registro onde o provedor de serviços pode obter, a partir da descrição do serviço, o mecanismo para vínculo com o serviço Web do provedor de serviços, podendo então chamar esse serviço Web (HENDRICKS, 2002, p. 10).

Um registro de serviços é a localização central onde o provedor de serviços pode relacionar seus serviços Web, e no qual um consumidor de serviços pode pesquisar serviços Web. Os provedores de serviços normalmente publicam sua capacidade de serviços Web no

registro do serviço para que, dessa forma, os consumidores de serviço os encontrem e, em seguida, vinculem-se ao seu serviço Web (HENDRICKS, 2002, p. 10).

Podemos identificar também três operações fundamentais para o funcionamento dos serviços Web: publicação, localização e vínculo. Para obtermos uma comunicação entre as aplicações, desconsiderando o tipo de linguagem e a plataforma onde será executada. Para que isso ocorra, precisamos dos padrões de cada uma dessas três operações e de uma maneira padrão para um provedor de serviços, visando descrever seu serviço Web (HENDRICKS, 2002, p. 10).

De acordo com Camelo (2002, p.6), “A comunicação entre aplicações de *Web Services* fazem uso de quatro camadas que empacotam a requisição e a resposta entre um servidor e um cliente”. As camadas utilizadas são assim dispostas:

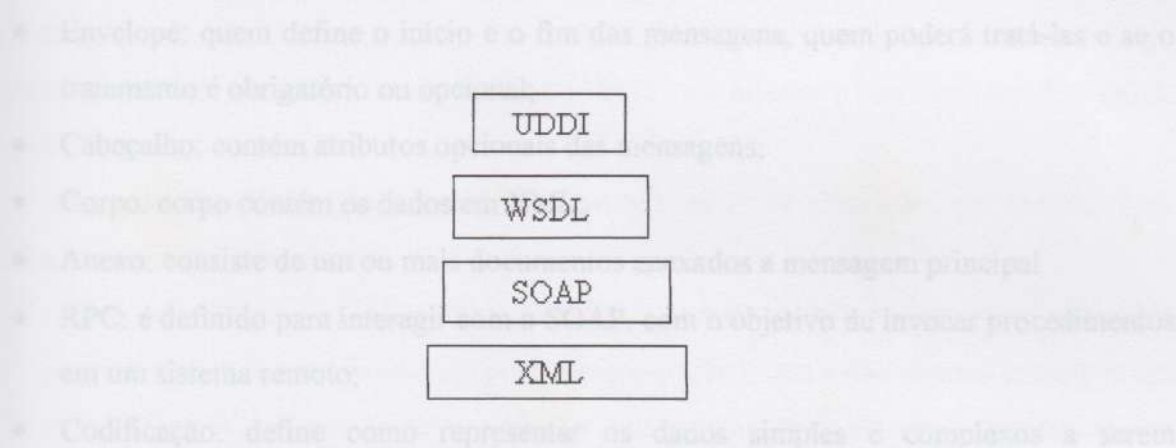


Figura 6 - Sobreposição de Camadas de Comunicação de um Web Service
Fonte: Hendricks (2002)

2.3.1 SOAP

Segundo Cunha (2002):

SOAP é um protocolo projetado para invocar aplicações remotas através de RPC (*Remote Procedure Calls* - Chamadas Remotas de Procedimento) ou trocas de mensagens, em um ambiente independente de plataforma e linguagem de programação. SOAP é, portanto, um padrão normalmente aceita para utilizar-se com *Web Services*. Desta forma, pretende-se garantir a interoperabilidade e intercomunicação entre diferentes sistemas, através da utilização de uma linguagem (XML) e mecanismo de transporte (HTTP) padrão.

De acordo com Hendricks (2002, p.12), para trocar informações em ambiente descentralizado e distribuído que permite comunicação entre aplicações de forma simples e

completamente independente de sistema operacional, linguagem de programação ou plataforma o serviço Web utiliza o protocolo SOAP (*Simple Object Access Protocol*).

A comunicação é realizada através de trocas de mensagens, transmitidas em formato XML, incluindo os parâmetros usados na chamada, bem como os dados de resultados, podendo também, ser utilizado para invocar, publicar e localizar os serviços Web no registro UDDI (*Universal Description, Discovery, and Integration*), combinar com uma variedade de outros protocolos, como HTTP, SMTP (*Simple Mail Transfer Protocol*), FTP (*File Transfer Protocol*), dentre outros, suportando, inclusive RPC (*Remote Procedure Call*).

O modelo de dados SOAP oferece definições para tipos de dados como *string*, *integer*, *float*, *double* e *date*.

Um pacote SOAP possui as seguintes partes:

- Envelope: quem define o início e o fim das mensagens, quem poderá tratá-las e se o tratamento é obrigatório ou opcional;
- Cabeçalho: contém atributos opcionais das mensagens;
- Corpo: corpo contém os dados em XML;
- Anexo: consiste de um ou mais documentos anexados a mensagem principal
- RPC: é definido para interagir com o SOAP, com o objetivo de invocar procedimentos em um sistema remoto;
- Codificação: define como representar os dados simples e complexos a serem transmitidos nas mensagens.

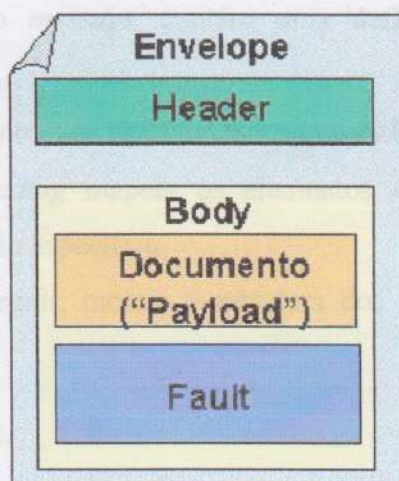


Figura 7 - Exemplo da Estrutura de Uma Mensagem SOAP
Fonte: Cunha (2002)

A figura 7, acima, mostra como pode ser montada uma estrutura SOAP.

Os serviços Web também podem ser utilizados para implementar arquiteturas orientadas a serviços, as *Service-Oriented Architectures* (SOA). Neste modelo de arquitetura os principais requisitos viram serviços e são acessados por outros serviços, modularizando e aumentando a coesão dos componentes da aplicação.

2.3.2 WSDL

Segundo Souza (2005, p.10), “A WSDL (*Web Service Description Language*) é um padrão que utiliza o formato (linguagem) XML para descrever interfaces Web”.

De acordo com Camelo (2002, p.8), “O WSDL descreve os serviços disponibilizados à rede através de uma semântica XML, este provê a documentação necessária para se chamar um sistema distribuído e o procedimento necessário para que esta comunicação se estabeleça. Enquanto o SOAP especifica a comunicação entre um cliente e um servidor O WSDL descreve os serviços oferecidos”.

Segundo Hendricks (2002, p.14), Em um alto nível de abstração, um serviço Web contém os seguintes elementos:

- Tipos de Porta: os elementos *portType* contêm um conjunto de operações representadas como elementos *operation* (operação), que estão apenas presentes em um serviço Web. Uma operação pode, ter uma mensagem de entrada e uma mensagem de saída, ou apenas uma mensagem de entrada ou saída;
- Mensagens: um elemento *message* contém uma definição dos dados a serem transmitidos;
- Tipos: o elemento *types* contém os tipos de dados que estão presentes na mensagem;
- Vínculos: o elemento *binding* mapeia os elementos *operation* em um elemento *portType*, para um protocolo específico.

A figura 8, mostrada a seguir, mostra a estrutura dos principais elementos de um WSDL.

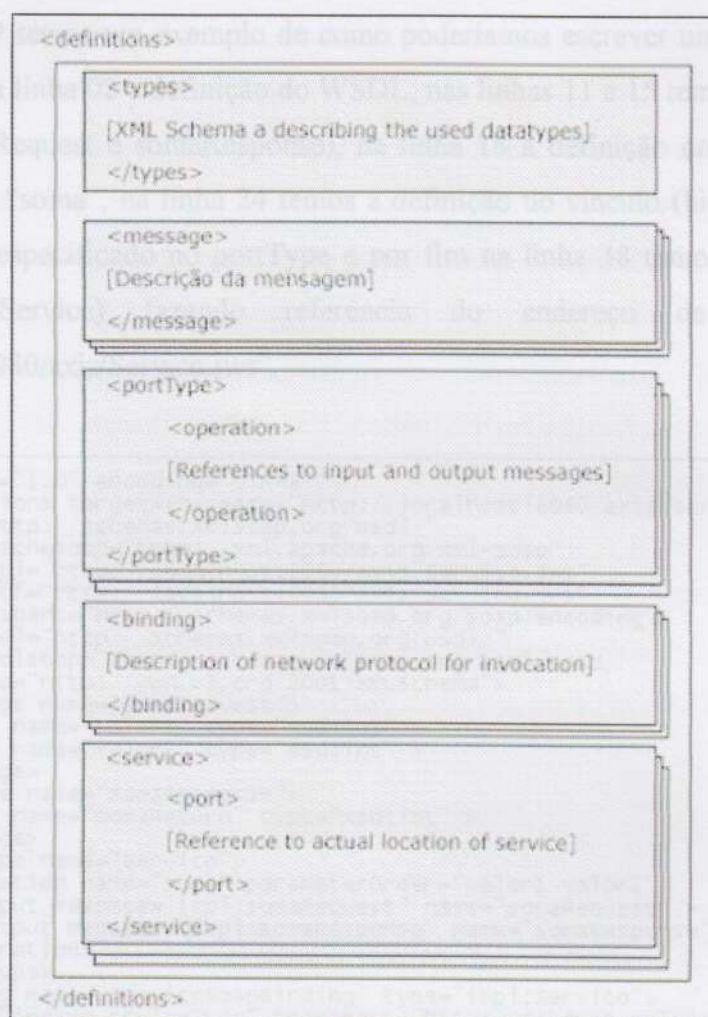


Figura 8 - Exemplo dos Principais Elementos de um WSDL
 Fonte: Hendricks (2002)

Basicamente ela define os métodos que estão presentes no serviço Web, os parâmetros de entrada/saída para cada um dos métodos, os tipos de dados, o protocolo de transporte usado e a URL da extremidade onde o serviço Web será hospedado. A XML não é apenas utilizada como formato para troca de mensagens, mas também como a forma através da quais os serviços são definidos.

Para permitir que os provedores de serviços publiquem detalhes sobre suas empresas e os serviços Web fornecidos em um registro central e que os consumidores de serviços localizem os provedores de serviços e detalhes sobre seus serviços Web, utiliza-se como padrão a UDDI (*Universal Description, Discover, and Integration*). Para descrever, descobrir e integrar serviços Web, existe um registro global público, chamado UDDI *business registry*, no qual toda a informação está disponível para consultas em geral. Entretanto, um registro privado pode adicionar controle de segurança para proteger os dados e prevenir acessos não autorizados.

Na figura 9 temos um exemplo de como poderíamos escrever um código de WSDL. Nesse caso logo na linha 02 a definição do WSDL, nas linhas 11 e 15 temos as definições das mensagens (somaRequest e somaResponse), na linha 18 a definição do portType (servico) com sua operação “soma”, na linha 24 temos a definição do vínculo (binding) mapeando a operação “soma” especificado no portType e por fim na linha 38 temos a identificação do serviço (ServicoService) fazendo referência do endereço de sua localização “http://localhost:8080/axis/Servico.jws”.

```

01. <?xml version="1.0" encoding="UTF-8"?>
02. <wSDL:definitions targetNamespace="http://localhost:8080/axis/Servico.jws"
03.     xmlns="http://schemas.xmlsoap.org/wSDL/"
04.     xmlns:apacheSOAP="http://xml.apache.org/xml-soap"
05.     xmlns:impl="http://localhost:8080/axis/Servico.jws"
06.     xmlns:intf="http://localhost:8080/axis/Servico.jws"
07.     xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
08.     xmlns:wSDL="http://schemas.xmlsoap.org/wSDL/"
09.     xmlns:wSDLsoap="http://schemas.xmlsoap.org/wSDL/soap/"
10.     xmlns:xsd="http://www.w3.org/2001/XMLSchema">
11.     <wSDL:message name="somaRequest">
12.         <wSDL:part name="valor1" type="xsd:int"/>
13.         <wSDL:part name="valor2" type="xsd:int"/>
14.     </wSDL:message>
15.     <wSDL:message name="somaResponse">
16.         <wSDL:part name="somaReturn" type="xsd:int"/>
17.     </wSDL:message>
18.     <wSDL:portType name="servico">
19.         <wSDL:operation name="soma" parameterOrder="valor1 valor2">
20.             <wSDL:input message="impl:somaRequest" name="somaRequest"/>
21.             <wSDL:output message="impl:somaResponse" name="somaResponse"/>
22.         </wSDL:operation>
23.     </wSDL:portType>
24.     <wSDL:binding name="ServicoSOAPBinding" type="impl:servico">
25.         <wSDLsoap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
26.         <wSDL:operation name="soma">
27.             <wSDLsoap:operation soapAction="">
28.                 <wSDL:input name="somaRequest">
29.                     <wSDLsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
30.                         namespace="http://DefaultNamespace" use="encoded"/>
31.                 </wSDL:input>
32.                 <wSDL:output name="somaResponse">
33.                     <wSDLsoap:body encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
34.                         namespace="http://localhost:8080/axis/Servico.jws" use="encoded"/>
35.                 </wSDL:output>
36.             </wSDL:operation>
37.         </wSDL:binding>
38.     <wSDL:service name="ServicoService">
39.         <wSDL:port binding="impl:ServicoSOAPBinding" name="servico">
40.             <wSDLsoap:address location="http://localhost:8080/axis/Servico.jws"/>
41.         </wSDL:port>
42.     </wSDL:service>
43. </wSDL:definitions>

```

Figura 9 - Exemplo de Código WSDL

Fonte: Pamplona (2004)

2.3.3 UDDI

De acordo com Camelo (2002, p.8), “UDDI é um padrão desenvolvido para prover informações em forma de diretório de seus negócios e seus *Web Services*. Este representa um

serviço *broker* que habilita as requisições a serviços para que seja encontrado um provedor do serviço”.

Segundo Souza (2005, p.17), “UDDI é uma especificação técnica para descrever, descobrir e integrar *Web Services*. Para que um serviço seja utilizado é necessário que o cliente consiga localizá-lo, e esta localização pode ser feita através do UDDI”.

Já Hendricks (2002, p.174) diz que, “UDDI é, principalmente, um registro comercial que permite que os negócios registrados armazenem mais informações sobre os serviços por eles oferecidos, e quais especificações eles aceitam. Essas especificações são representadas pela noção de um *tModel*, que atua como um recipiente para qualquer tipo de informação técnica”.

De acordo Hendricks (2002, p.145), “*tModel* representa uma especificação técnica em um registro UDDI. Essa especificação técnica pode descrever todos os tipos de coisas – por exemplo, um padrão para a troca de dados entre negócios ou a definição da interface de um determinado serviço Web”.

2.4 JAVA

Segundo Deitel (2001, p.56), Java é uma poderosa linguagem de programação de computador desafiadora, informativa e divertida. Aprimorada para programadores experientes construírem poderosos sistemas de informações. É portátil, por isso pode ser executado em muitos computadores diferentes e completamente orientado a objeto com um forte suporte técnico adequada de engenharia de software.

A linguagem de programação Java vem sendo cada vez mais utilizada em dispositivos móveis. Ela torna-se importante por ser uma linguagem portátil, orientada a objeto e ideal para dispositivos com pouca memória (DEITEL, p.57).

De acordo com Deitel (2001, p.64), “Muitas pessoas acreditam que a próxima área importante em que os microprocessadores terão um impacto profundo será os dispositivos eletrônicos inteligentes destinados ao consumidor final. Reconhecendo isso, a *Sum Microsystems* financiou uma pesquisa corporativa interna com o codinome Green em 1991”.

Segundo Gosling (2003) apud Costa et al, a linguagem Java foi criada através da linguagem C++. Programadores que utilizavam essa linguagem retiraram todas as partes ruins da linguagem C++ e adicionaram outras funções que a deixava muito mais poderosa e útil, criaram a OAK, que se tornou leve, segura e portátil.

A Sun transformou o *Green Team* em uma empresa chamada *First Person* que tinha um conceito muito interessante, a linguagem Oak, mas não tinham idéia do que fazer com ele. Mais tarde foi descoberto que o nome Oak já era usado, então a equipe mudou o nome para JavaTM, nome de uma cafeteria que vendia café importado e que também é uma gíria americana para “cafezinho”. Porém na época, entre 1993 e 1994 houve a explosão da Internet e da *World Wide Web* e a SunTM viu uma ótima oportunidade para a linguagem JavaTM, que permitia construção de página totalmente dinâmicas, leves, que funcionava bem nos *browsers* e que era um grande concorrente para as páginas estáticas construídas em HTML. Isso deu um novo rumo e incentivo à linguagem, Golsling (apud Costa et al, 2003).

Segundo Deitel (2001, p.59), “foi em 1995, que a Sun anunciou Java formalmente em uma conferência importante. Normalmente, um evento como esse não teria gerado muita atenção. Entretanto gerou interesse imediato na comunidade comercial por causa do fenomenal interesse pela *World Wide Web*”.

Schaefer (2004) diz que: “Os programas desenvolvidos com a linguagem Java podem ser interpretados e executados em qualquer sistema através de um interpretador Java, ou *Java Virtual Machine* (JVM)”.

Já Rocha (apud Dias; Fontes, 2005): “A máquina virtual Java é um computador hipotético, implementado como uma aplicação de software em uma máquina real”. Uma aplicação desenvolvida em Java onde pode utilizar um conjunto de bibliotecas nativas que é conhecido como *Java Application Programming Interface* (API). Juntas, a JVM (*Java Virtual Machine*) e Java API constituem o ambiente de execução de Java ou ainda, a plataforma Java.

De acordo com Dias; Fontes (2005) apud Silva:

A API Java invoca métodos nativo. Existem os dois métodos: Java e Nativo. Um método Java é compilado para *bytecodes*, e armazenado em um arquivo.class e o método Nativo é escrito em alguma outra linguagem, tal como C, C++ ou *assembly*, e compilado para código nativo de máquina de um processador particular. Os métodos nativos não são. Quando um programa Java chama um método nativo, a JVM carrega a biblioteca de vínculo dinâmico que contém o método nativo e o invoca.

2.4.1 Principais Características do Java

Java é uma linguagem de alto nível, com sintaxe extremamente similar à do C++, e com diversas características herdadas de outras linguagens, como Smalltalk e Modulo-3. É antes de tudo uma linguagem simples, independente de arquitetura, robusta, segura,

extensível, bem estruturada, distribuída, *multithreaded* e com *garbage collection*, (DEITEL, 2001, p.64).

Segundo Deitel (2001, p.64):

Java é muito parecida com C++, mas muito mais simples. Java não possui sobrecarga de operadores, *structs*, *unions*, aritmética de ponteiros, herança múltipla, diretivas de pré-processamento e a memória alocada dinamicamente são gerenciadas pela própria linguagem, que usa algoritmos de *garbage collection* para desalocar regiões de memória que não estão mais em uso.

Deitel (2001, p. 67) diz que:

Ao contrário de C++, que é uma linguagem híbrida, Java é uma linguagem orientada a objetos que segue a linha purista iniciada por *Smalltalk*. Com a exceção dos tipos básicos da linguagem (*int*, *float*, etc.), a maior parte dos elementos de um programa Java são objetos. O código é organizado em classes, que podem estabelecer relacionamentos de herança simples entre si, somente a herança simples é permitida em Java.

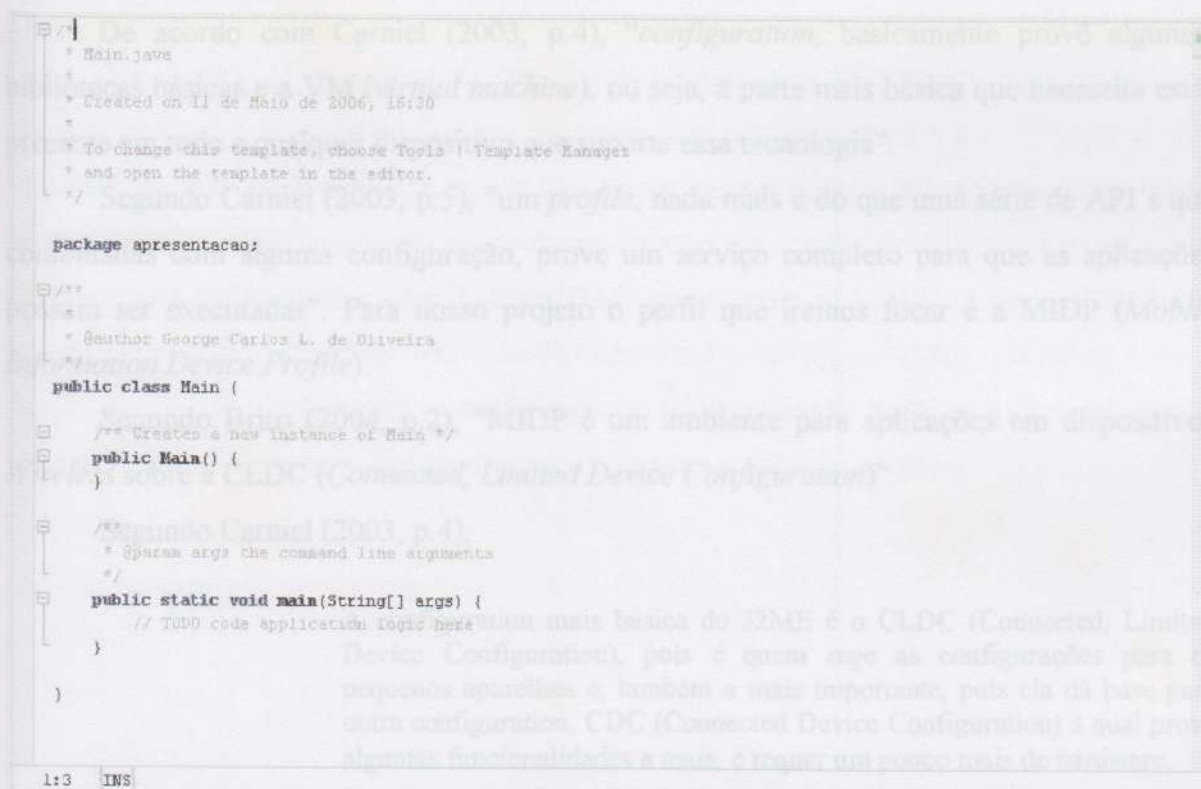
Java tem processamento distribuído devido às suas chamadas a funções de acesso remoto (*sockets*) e aos protocolos de Internet mais comuns (HTTP, FTP, Telnet, etc.) desta forma a elaboração de aplicativos baseados em arquiteturas cliente-servidor é facilmente obtida (DEITEL, 2001, p.891).

As maiores partes dos sistemas operacionais hoje no mercado dão suporte à *multithreading* (multitarefa). Java pode possuir mais de uma linha de execução (*thread*). Por exemplo, cálculos extensos, que em geral demandam muito tempo do processador, podem ser escritos em uma *thread*, e a parte de interface com o usuário, que depende mais dos periféricos de I/O que do processador, pode ser executada em outra *thread* (DEITEL, 2001, p.240).

Como a máquina virtual Java faz uma verificação em tempo de execução quanto aos acessos de memória, abertura de arquivos e uma série de eventos que podem gerar um travamento em outras linguagens, mas que em programas Java são geradas exceções. Em geral, ao escrever programas Java utilizando-se de herança de classes predefinidas, o programador é quase que forçado a escrever algumas rotinas de tratamento de exceção, um trabalho que, se de início pode parecer forçado, irá poupar bastante dor de cabeça no futuro (DEITEL, 2001, p.373).

Uma das características de Java que a tornou ideal para seu uso na elaboração de aplicativos distribuídos foi a sua independência de plataforma, *Machine Independent*. Afinal, a Internet é uma grande “salada” de computadores de todos os tipos, dotados dos mais diversos sistemas operacionais e ambientes gráficos que se possa imaginar. Java consegue essa independência devido ao fato de que o compilador Java não gera instruções específicas a uma plataforma, mas sim um programa em um código intermediário, denominado *bytecodes*, que pode ser descrito como uma linguagem de máquina destinada a um processador virtual que não existe fisicamente (DEITEL, 2001, p.72).

Na figura 10 temos um exemplo de código Java bem simples e básico, contendo um pacote “package apresentacao” e uma classe “public class Main”. A classe sendo composta por dois métodos um público e outro estático, representados respectivamente por “public Main()” e “public static void main(String[] args)”.



```

Main.java
Created on 11 de Maio de 2006, 16:30
To change this template, choose Tools | Template Manager
and open the template in the editor.
package apresentacao;

/**
 *
 * @author George Carlos L. de Oliveira
 */
public class Main {

    /** Creates a new instance of Main */
    public Main() {
    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        // TODO code application logic here
    }

}

```

Figura 10 - Exemplo de Código em Java
Fonte: O próprio autor

2.5 J2ME

Aplicações voltadas para computadores móveis como telefones celulares são cada vez mais demandadas. Recentes avanços na tecnologia empregada em tais dispositivos

contribuíram para diminuir o custo dos mesmos e torná-los acessíveis a um grupo maior de pessoas. Esses avanços também contribuíram para aumentar a capacidade computacional de dispositivos móveis, de modo que, atualmente são capazes de utilizar informações disponíveis na Internet ou que podem comportar-se como clientes em aplicações de comércio eletrônico, (PEREIRA, 2004, p.21).

O J2ME é uma plataforma que foi desenvolvida para que os desenvolvedores consigam fazer aplicativos para dispositivos com pouca memória, com telas pequenas e poder de processamento limitado, o que inclui telefones celulares, PDAs e *paggers*.

Segundo Oliveira (2006, p. 27), “J2ME é uma API Java voltada para micro aplicativos que rodam em micro processadores assim como os dos celulares, palms, [PDA’s, *paggers*, etc.]”.

De acordo com Carniel (2003, p.5), “J2ME basicamente é dividido em: *configurations* (configuração), *profiles* (perfil) e API’s convencionais”.

De acordo com Carniel (2003, p.4), “*configuration*, basicamente provê algumas bibliotecas básicas e a VM (*virtual machine*), ou seja, a parte mais básica que necessita estar presente em todo e qualquer dispositivo que suporta essa tecnologia”.

Segundo Carniel (2003, p.5), “um *profile*, nada mais é do que uma série de API’s que combinadas com alguma configuração, prove um serviço completo para que as aplicações possam ser executadas”. Para nosso projeto o perfil que iremos focar é a MIDP (*Mobile Information Device Profile*).

Segundo Brito (2004, p.2), “MIDP é um ambiente para aplicações em dispositivos *Wireless* sobre a CLDC (*Connected, Limited Device Configuration*)”.

Segundo Carniel (2003, p.4),

A configuration mais básica do J2ME é o CLDC (Connected, Limited Device Configuration), pois é quem rege as configurações para os pequenos aparelhos e, também a mais importante, pois ela dá base para outra configuration, CDC (Connected Device Configuration) a qual provê algumas funcionalidades a mais, e requer um pouco mais de hardware.

Segundo Orrico (2006):

Uma aplicação móvel escrita em Java é chamada de *Midlet* (MID vem de *Mobile Information Device*, ou seja, Dispositivos de Informação Móveis). A classe principal de cada aplicação móvel, portanto, deve estender a classe abstrata *MIDlet*, pois é esta classe que oferece os principais métodos para controlar o ciclo de vida das aplicações.

De acordo com Orrico (2006), As aplicações móveis possuem três estados que representam o ciclo de vida das aplicações móveis: *startApp* (iniciado), *pauseApp* (pausado) e *destroyApp* (destruído).

- *startApp()* (iniciado): momento em que as aplicações começam a rodar;
- *pauseApp()* (pausado): entram no estado de pausa. Normalmente, as aplicações entram num estado de pausa quando o dispositivo móvel recebe um chamado de algum evento com maior prioridade do que a aplicação em execução. Além disso, uma aplicação pode entrar no estado de pausa a pedido do usuário;
- *destroyApp()* (finalizado): este estado destruído é acionado quando a aplicação é finalizada, fato que pode ocorrer à 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.

Na figura 11 é mostrado um código Java para dispositivo móvel (*Midlet*) comunicando-se com um *Web Service*, passando dois números inteiros e mostrando o valor retornado na tela. Podemos notar que o *MIDlet* não possui método *main()* para rodar. Nesse pequeno código vemos a importação de quatro pacotes, nas quatro primeiras linhas, na quinta linha vemos a definição da classe “*ClienteJ2ME*” que estende de *MIDlet*. A classe é composta por quatro métodos, *startApp* (tem a função de iniciar a aplicação J2ME), *pauseApp* (com a função de pausar a aplicação) e *testWebService* (com a função de fazer a conexão com o serviço Web passar os dois inteiros digitados e mostrar o valor de retorno).

Jeveaux (2003) afirma que: “A linguagem Java encontrou um meio de manter as suas características básicas, desenvolvendo uma tecnologia para acesso a banco de dados independente de plataforma e independente de fornecedor”.

Segundo Jevaux (2003),

Devido a enorme quantidade de bancos de dados do mercado com linguagens proprietárias, os projetistas do JavaSoft resolveram criar uma API para acesso a banco de dados padrão SQL (*Structure Query Language*), ou seja, uma interface entre a linguagem Java e outras linguagens que todos os bancos de dados suportam. A linguagem escolhida é a SQL, então a fabricante de bancos de dados fornecem os drivers para SQL e Java cria a API chamada de JDBC (*Java Data Base Connectivity*).

```

import javax.microedition.lcdui.Display;
import javax.microedition.lcdui.TextBox;

import org.ksoap.soapObject;
import org.ksoap.transport.HttpTransport;

public class clienteJ2ME extends javax.microedition.midlet.MIDlet {
    private Display display;
    private String url = "http://localhost:8080/axis/servico.jws";
    TextBox textBox = null;

    public void startApp() {
        display = Display.getDisplay(this);
        try {
            testWebService();
        } catch (Exception ex) {
            System.out.println(ex);
        }
    }

    public void pauseApp() {}
    public void destroyApp(boolean unconditional) {}

    public void testWebService() throws Exception {
        StringBuffer stringBuffer = new StringBuffer();
        TextBox textBox = null;

        // chama o webservice
        soapObject client = new soapObject(url, "soma");
        client.addProperty("valor1", new Integer(2));
        client.addProperty("valor2", new Integer(4));
        HttpTransport ht = new HttpTransport(url, "soma");
        stringBuffer.append("Resultado: " + ht.call(client));

        // mostra o valor do resultado na tela.
        textBox = new TextBox("Teste webservice", stringBuffer.toString(), 1024, 0);
        display.setCurrent(textBox);
    }
}

```

Figura 11 - Exemplo de Código J2ME
Fonte: O próprio autor

2.6 JDBC

Jeveaux (2003) afirma que: “A linguagem Java encontrou um meio de manter as suas características básicas, desenvolvendo uma tecnologia para acesso a banco de dados independente de plataforma e independente de fornecedor”.

Segundo Jeveaux (2003),

Devido a enorme quantidade de bancos de dados do mercado com linguagens proprietárias, os projetistas do JavaSoft resolveram criar uma API para acesso a banco de dados padrão SQL (*Structure Query Language*), ou seja, uma interface entre a linguagem Java e outra linguagem que todos os bancos de dados suportam. A linguagem escolhida é a SQL, então a fabricante de bancos de dados fornecem os *drivers* para SQL e Java cria a API chamada de JDBC (*Java Daba Base Connectivity*).

A figura 12 ilustra a tecnologia do JDBC, onde os programadores Java devem acessar um banco de dados efetuando declarações SQL padrão, através de uma API JDBC. Estas declarações são enviadas ao gerenciador de *drivers* JDBC, que se encarrega de utilizar os *drivers* disponíveis para o banco de dados. Quando já está implementado um *driver* JDBC para o determinado banco de dados, o gerenciador passa as declarações SQL para o *driver* através de uma API do *driver* JDBC. Caso não tenha um *driver* JDBC disponível, então é utilizada uma ponte JDBC/ODBC, que envia as declarações SQL para o *driver* ODBC (interface de programação C para SQL).

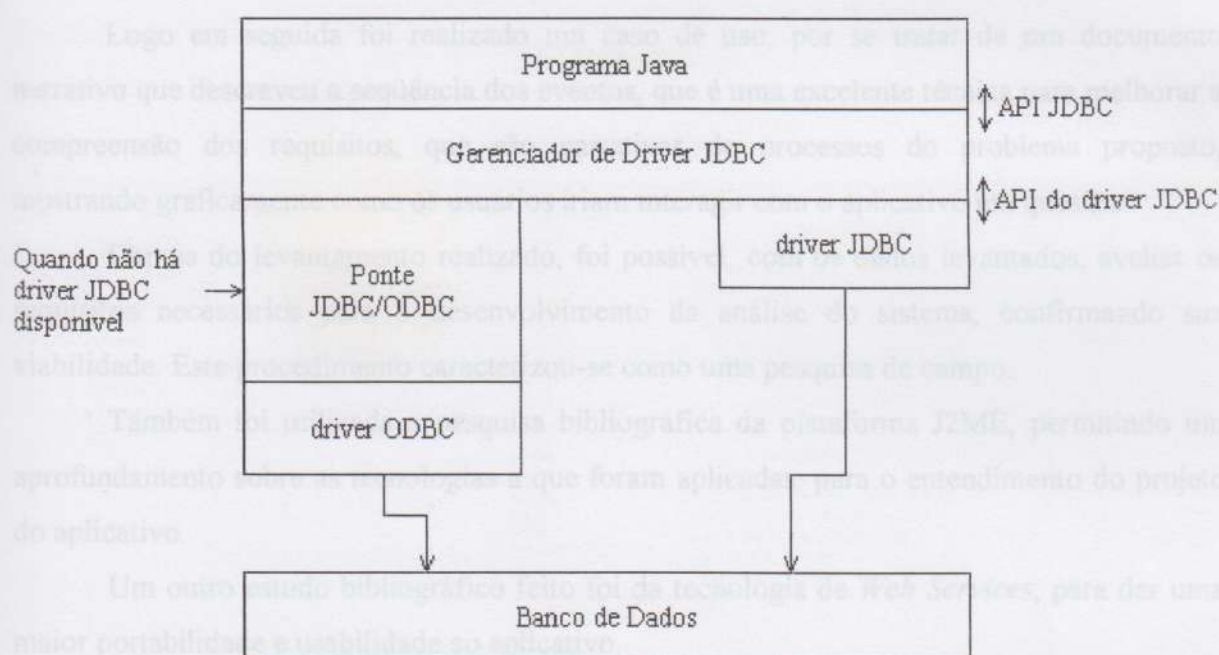


Figura 12 - Exemplo da Estrutura do JDBC

Fonte: Jeveaux (2003)

3 METODOLOGIA

Para elaboração deste projeto, foi realizado inicialmente um levantamento das informações necessárias para verificar quais os dados que o banco de dados, simulado, deveria ter, para que o aplicativo pudesse dar as respostas proposta por este projeto.

Como passo seguinte, foi feito uma especificação dos requisitos descrevendo as funcionalidades e necessidades do aplicativo, por ter como objetivo básico a identificação e a documentação do que foi realmente necessário, em uma forma que comunicasse claramente essa informação ao usuário e a equipe de desenvolvimento.

Logo em seguida foi realizado um caso de uso, por se tratar de um documento narrativo que descreveu a seqüência dos eventos, que é uma excelente técnica para melhorar a compreensão dos requisitos, que são narrativas de processos do problema proposto, mostrando graficamente como os usuários iriam interagir com o aplicativo em questão.

Depois do levantamento realizado, foi possível, com os dados levantados, avaliar os requisitos necessários para o desenvolvimento da análise do sistema, confirmando sua viabilidade. Este procedimento caracterizou-se como uma pesquisa de campo.

Também foi utilizada a pesquisa bibliográfica da plataforma J2ME, permitindo um aprofundamento sobre as tecnologias a que foram aplicadas, para o entendimento do projeto do aplicativo.

Um outro estudo bibliográfico feito foi da tecnologia de *Web Services*, para dar uma maior portabilidade e usabilidade ao aplicativo.

Concluído o estudo bibliográfico, foi criado um documento de requisitos que serviu de base para o início da análise do sistema. Fazendo uso da tecnologia de *Web Services* permitindo a criação dos diagramas de casos de uso e de colaboração.

O passo seguinte foi realizado em etapas, com o objetivo da criação de um aplicativo.

As etapas são identificadas como:

- Definição das classes e definição e criação do banco de dados;
- Projeto da utilização de *Web Services*.
- Validação da utilização da linguagem de programação o Java, mais especificamente a plataforma J2ME.

A finalidade da criação deste projeto foi de avaliar a funcionalidade e a portabilidade de um aplicativo para dispositivos móveis. Esses procedimentos caracterizaram-se como técnica de pesquisa de laboratório.

4 RESULTADO OBTIDO

4.1 DESENVOLVIMENTO DE APLICAÇÕES COM J2ME

Este capítulo tem por objetivo demonstrar como construir uma aplicação bem simples para plataforma J2ME. Para isso, foi utilizado o programa Netbeans IDE 5.0, no menu principal opção *File->New Project* ou digitar as teclas *CTRL+SHIFT+N*, esse procedimento se faz necessário para se criar o novo projeto, na janela que aparece escolhe-se, em *Categories* (categorias), escolher *Mobile* (móvel) e na janela ao lado *Projects* (Projeto), *Mobile Application* (Aplicação móvel), apertar o botão *Next* (próximo) para ir para janela seguinte, ver na figura 13, logo abaixo.

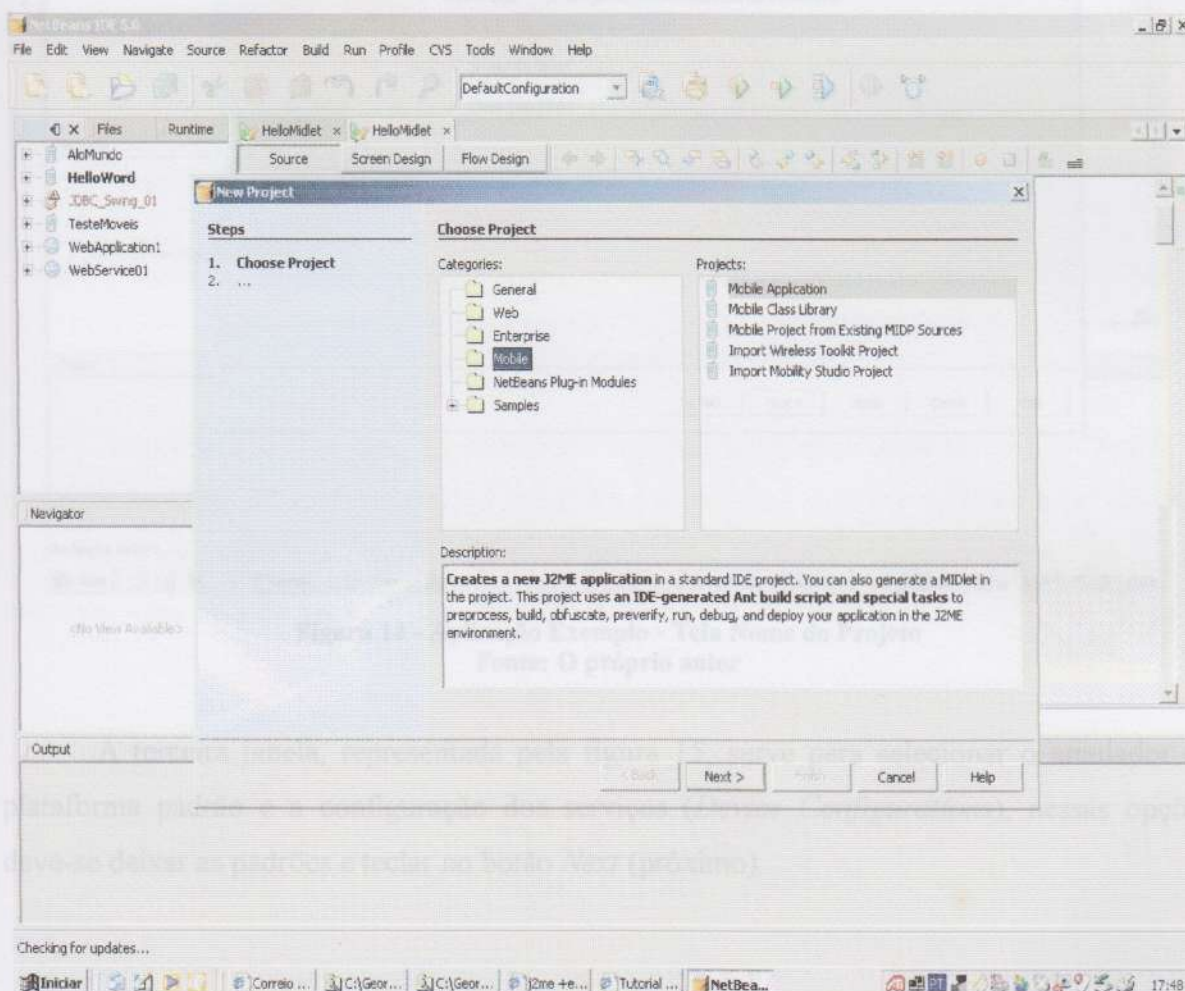


Figura 13 - Aplicação Exemplo - Tela Novo Projeto
Fonte: O próprio autor

A janela mostrada na figura 14 tem a finalidade de escolher o nome do projeto (*Project Name*), bem como sua localização (*Project Location*). Como este aplicativo só terá uma única classe e esta será a principal, e que, deverá ser iniciada quando o aplicativo for inicializado, devemos deixar as opções *Set as Main Project* e *Create Hello MIDlet* assinaladas ao final teclar no botão *Next* (próximo).

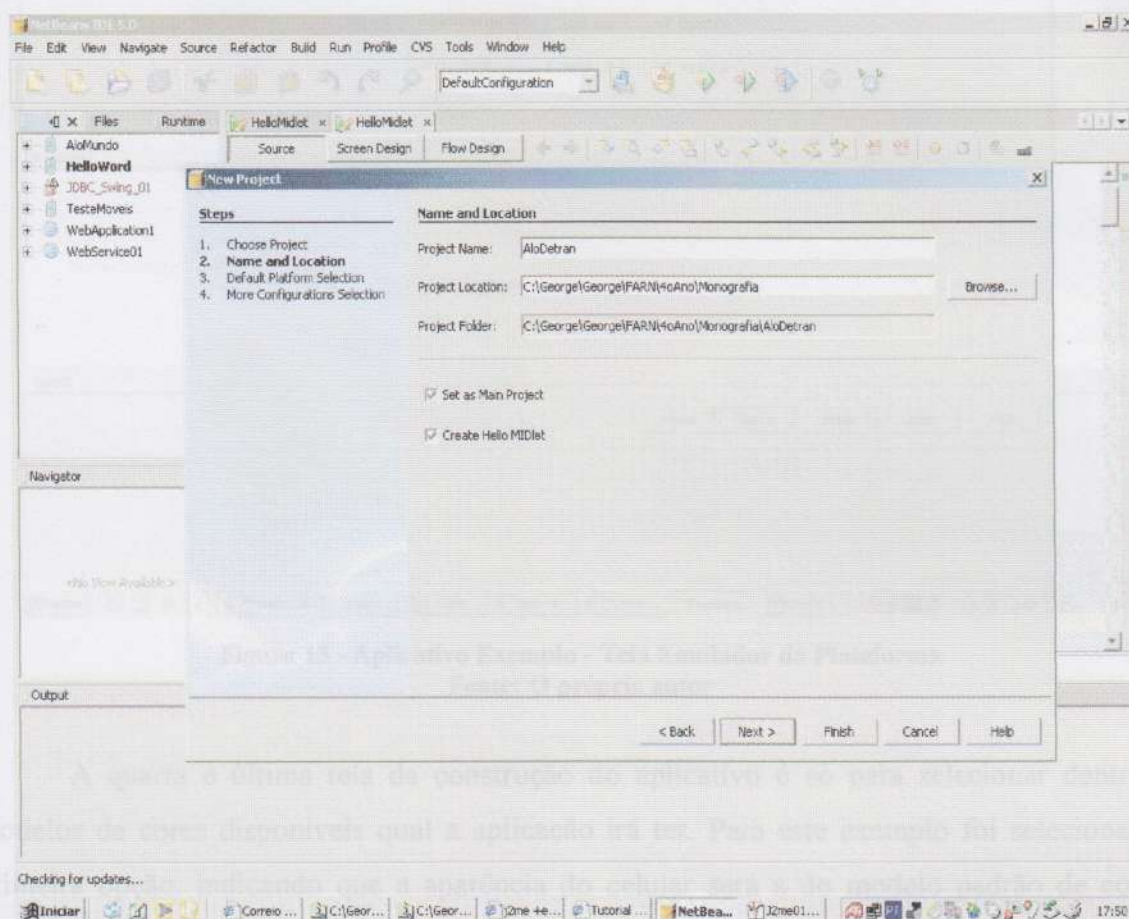


Figura 14 - Aplicação Exemplo - Tela Nome do Projeto
Fonte: O próprio autor

A terceira janela, representada pela figura 15, serve para selecionar o emulador da plataforma padrão e a configuração dos serviços (*Device Configurations*), nessas opções deve-se deixar as padrões e teclar no botão *Next* (próximo).

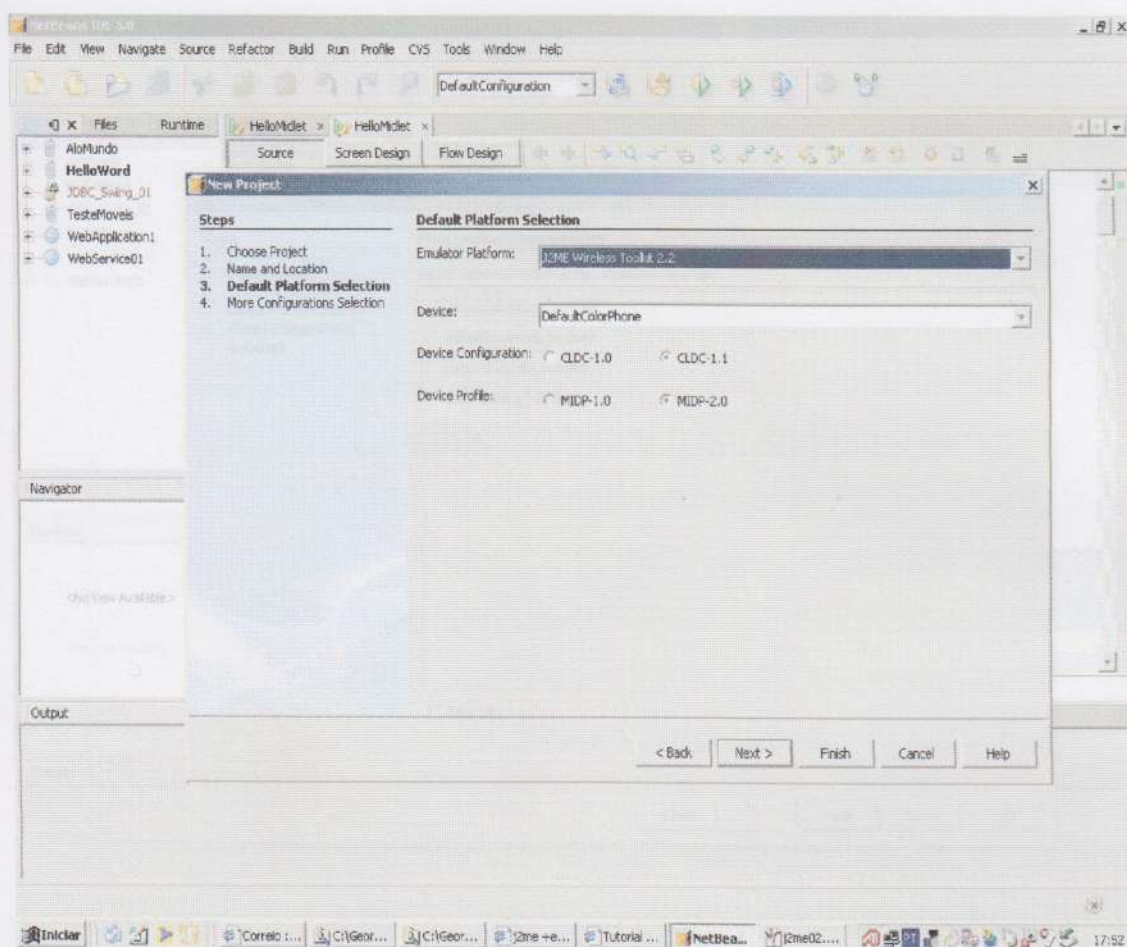


Figura 15 - Aplicativo Exemplo - Tela Emulador da Plataforma
Fonte: O próprio autor

A quarta e última tela da construção do aplicativo é só para selecionar dentre os modelos de cores disponíveis qual a aplicação irá ter. Para este exemplo foi selecionada a primeira opção, indicando que a aparência do celular será a do modelo padrão de cor do telefone. Veja a figura 16, logo abaixo.

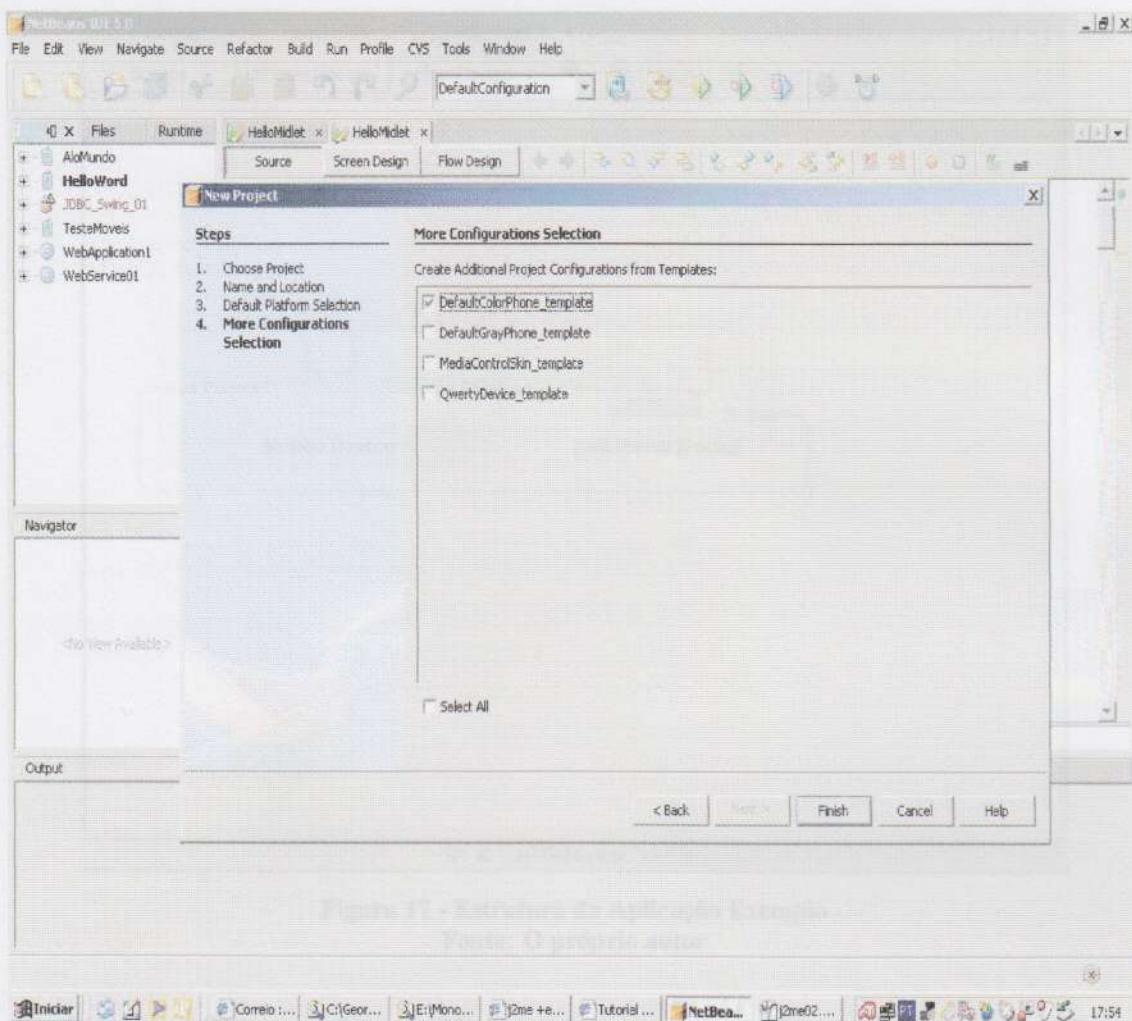


Figura 16 - Aplicativo Exemplo - Tela Modelo dos Celulares
Fonte: O próprio autor

A figura 17, abaixo, mostra a estrutura criada pelo assistente para nossa aplicação de exemplo, nela podemos observar que o aplicativo é iniciado pelo método “*Start Point*” e finalizado pelo método “*Sair*” e que é composto de apenas um formulário (*helloForm*).

```

package hello;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class HelloMidlet extends MIDlet implements CommandListener {

    /** Esta classe cria uma nova instância de HelloMidlet */
    public HelloMidlet() {
        // ...
    }

    private Form helloForm;
    private StringItem helloStringItem;
    private Command exitCommand;

    /** Este método tem a finalidade de inicializar a interface de usuário da
    aplicação. */
    private void initialize() {
        // ...
    }
}

```

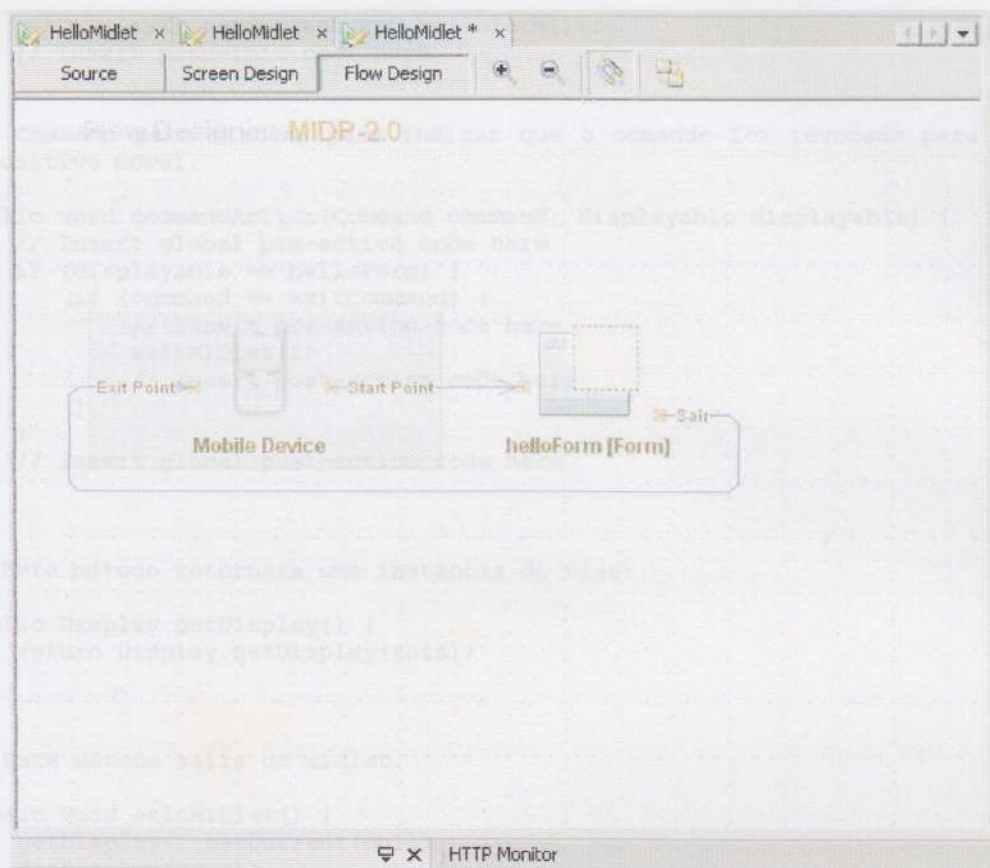


Figura 17 - Estrutura da Aplicação Exemplo
Fonte: O próprio autor

O Código fonte gerado pelo Netbeans encontra-se na figura 18 logo abaixo. Para um melhor entendimento o código fonte encontra-se todo comentado.

```

/*
 * HelloMidlet.java
 *
 * Created on 20 de Novembro de 2006, 17:54
 * Autor: George Carlos Lucena de Oliveira
 */

package hello;

import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class HelloMidlet extends MIDlet implements CommandListener {

    /** Este método cria uma nova instância de HelloMidlet */
    public HelloMidlet() {
    }

    private Form helloForm;
    private StringItem helloStringItem;
    private Command exitCommand;

    /** Este método tem a finalidade de inicializar a Interface do usuário da
    aplicação.
    */
    private void initialize() {
        // Insert pre-init code here
    }
}

```

```

        getDisplay().setCurrent(get_helloForm());
        // Insert post-init code here
    }

    /** Chamado pelo sistema para indicar que o comando foi invocado para o visor
    do dispositivo móvel.
    */
    public void commandAction(Command command, Displayable displayable) {
        // Insert global pre-action code here
        if (displayable == helloForm) {
            if (command == exitCommand) {
                // Insert pre-action code here
                exitMIDlet();
                // Insert post-action code here
            }
        }
        // Insert global post-action code here
    }

    /**
    * Este método retornara uma instância do visor.
    */
    public Display getDisplay() {
        return Display.getDisplay(this);
    }

    /**
    * Este método sairá do midlet.
    */
    public void exitMIDlet() {
        getDisplay().setCurrent(null);
        destroyApp(true);
        notifyDestroyed();
    }

    /** Este método retornará uma instância da classe "helloForm", sendo
    instanciado diretamente quando for inicializado. O valor de retorno é a própria
    instância do objeto.
    */
    public Form get_helloForm() {
        if (helloForm == null) {
            // Insert pre-init code here
            helloForm = new Form(null, new Item[] {get_helloStringItem()});
            helloForm.addCommand(get_exitCommand());
            helloForm.setCommandListener(this);
            // Insert post-init code here
        }
        return helloForm;
    }

    /** Este método retorna a mensagem do componente instanciado de helloStringItem
    que foi digitado diretamente no campo texto.
    */
    public StringItem get_helloStringItem() {
        if (helloStringItem == null) {
            // Insert pre-init code here
            helloStringItem = new StringItem("Exemplo de Aplica\u00E7\u00E3o J2ME",
            "Al\u00F4, Detran!");
            // Insert post-init code here
        }
        return helloStringItem;
    }

    /** Este método retorna uma instância do componente exitCommand diretamente
    para o campo, quando for instanciado.
    */
    public Command get_exitCommand() {
        if (exitCommand == null) {

```

```

// Insert pre-init code here
exitCommand = new Command("Sair", Command.EXIT, 1);
// Insert post-init code here
}
return exitCommand;
}
/* Método que tem por objetivo iniciar a aplicação */
public void startApp() {
    initialize();
}

/* Método que tem por objetivo parar a execução da aplicação */
public void pauseApp() {
}

/* Método que tem por objetivo destruir, finalizar, a aplicação */
public void destroyApp(boolean unconditional) {
}
}

```

Figura 18 - Código da Aplicação Alô Detran
Fonte: O próprio autor

4.2.1 Requisitos Funcionais

O software deve possibilitar ao usuário a consulta, via algum dispositivo móvel, de qualquer veículo, desde que este esteja cadastrado na Base de Dados do DETRAN e tenha uma placa que o identifique. Para que isso seja possível, este dispositivo móvel deverá poder fazer conexão com a Internet.

- O usuário deve digitar a placa do veículo, no celular, para consulta;
- O software faz a conexão, passando como parâmetro o número da placa digitada;
- O software deve retornar uma mensagem de nada consta ou de alguma infração e/ou impedimento;
- O software finaliza a conexão.

4.2.2 Requisitos não Funcionais

- Utilizar dispositivos móveis, servindo como aplicação cliente;
- Linguagem Java (J2ME) para o acesso;
- Persistir as informações em banco de dados;
- Utilizar o Webservice para padronizar o acesso ao serviço;
- Retornar o mínimo de informações para o usuário, com o intuito de agilizar o trabalho;
- O tempo de resposta não deverá ultrapassar 30 segundos;

4.2 ANÁLISE E PROJETO DO SISTEMA DE CONSULTA A PLACAS DE AUTOMÓVEIS APLICANDO AS TECNOLOGIAS DE WEBSERVICE E J2ME

O projeto proposto oferece ao desenvolvedor de sistemas uma visão de como o projeto poderá ser desenvolvido. Iniciando por um usuário qualquer que tenha um dispositivo móvel com acesso à Internet, este usuário deverá fazer a conexão com a Internet, abrir o sistema de consulta e digitar a placa do veículo que deverá ser consultado. O sistema deverá trazer como retorno de informação se o veículo pesquisado tem alguma infração, ou algum tipo de impedimento ou “Nada Consta”.

4.2.1 Requisitos Funcionais

O software deve possibilitar ao usuário a consulta, via algum dispositivo móvel, de qualquer veículo, desde que este esteja cadastrado na Base de Dados do DETRAN e tenha uma placa que o identifique. Para que isso seja possível, este dispositivo móvel deverá poder fazer conexão com a Internet.

- O usuário deve digitar a placa do veículo, no celular, para consulta;
- O software faz a conexão, passando como parâmetro o número da placa digitada;
- O software deve retornar uma mensagem de nada consta ou de alguma infração e/ou impedimento;
- O software finaliza a conexão.

4.2.2 Requisitos não Funcionais

- Utilizar dispositivos móveis, servindo como aplicação cliente;
- Linguagem Java (J2ME) para o acesso;
- Persistir as informações em banco de dados;
- Utilizar o *WebService* para padronizar o acesso ao serviço;
- Retornar o mínimo de informações para o usuário, com o intuito de otimizar o tráfego;
- O tempo de resposta não deverá ultrapassar 30 segundos;

4.2.3 Arquitetura do Sistema

A figura 19 logo abaixo, nos dá uma visão lógica e física do aplicativo. Ela mostra todos os componentes existentes para o desenvolvimento do aplicativo em questão. O componente “Celular” e “Servidor da Aplicação”, identificados pela cor verde são internos ao aplicativo, enquanto que os componentes “Gateway da Operadora” e “Servidor de Banco”, de cor amarela, são externos ao aplicativo.

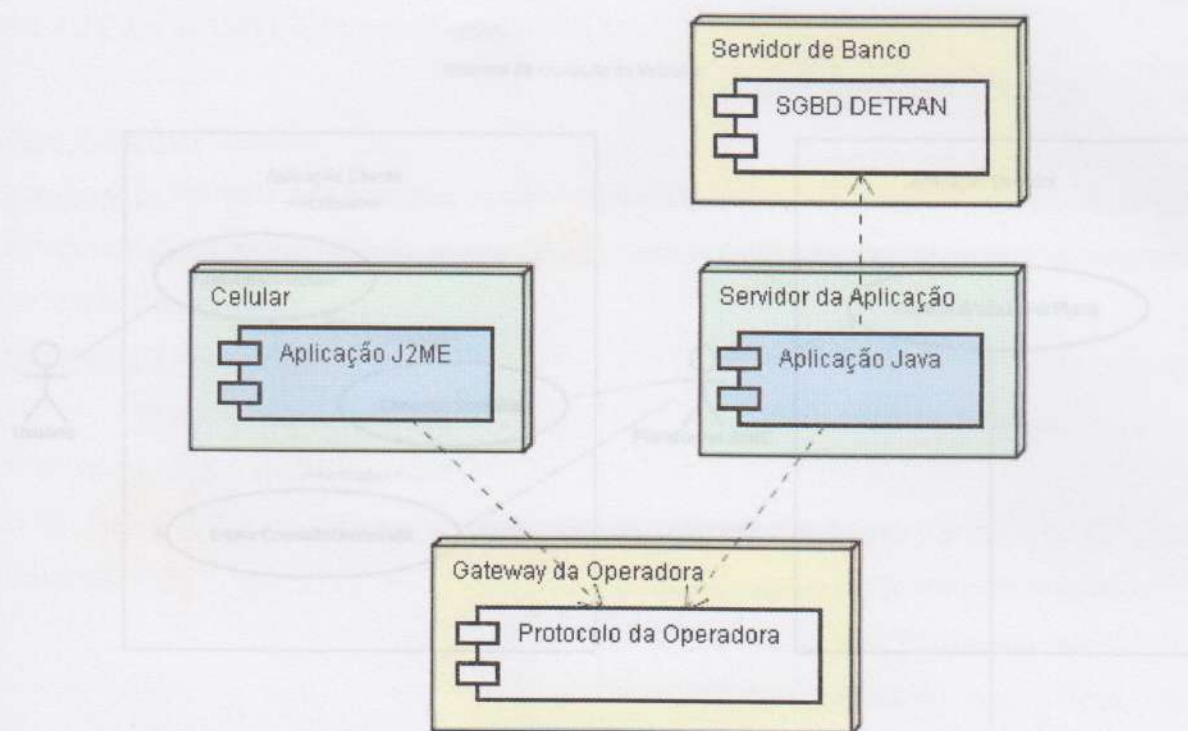


Figura 19 - Arquitetura Proposta para o Sistema
Fonte: O próprio autor

O componente “Celular” identifica que a aplicação é baseada na plataforma J2ME e que através de uma operadora, identificada pelo componente “Gateway da Operadora”, se comunica com o componente “Servidor da Aplicação” que é o próprio *Web Service* e que foi desenvolvido na linguagem Java, este é o responsável por fazer a ligação do banco de dados, representado pelo componente “Servidor de Banco” com o “Gateway da Operadora”.

A arquitetura da figura 19 acima mostra que o componente “Celular” tem uma dependência com o componente “Gateway da Operadora”, este com o “Servidor da Aplicação” que por sua vez tem uma dependência com “Servidor de Banco”.

4.2.4 Diagramas de Caso de Uso

A figura 20, como se vê logo abaixo faz referência ao caso de uso do sistema proposto, pois nos mostra um usuário qualquer, como ator, acessando uma aplicação cliente, representada por um celular, que faz uso da tecnologia J2ME como plataforma, representado por um outro ator, conectando-se a uma aplicação servidor, através de um serviço Web, acessando o banco de dados do DETRAN, como mostra o ator BancoDeDadosDETRAN.

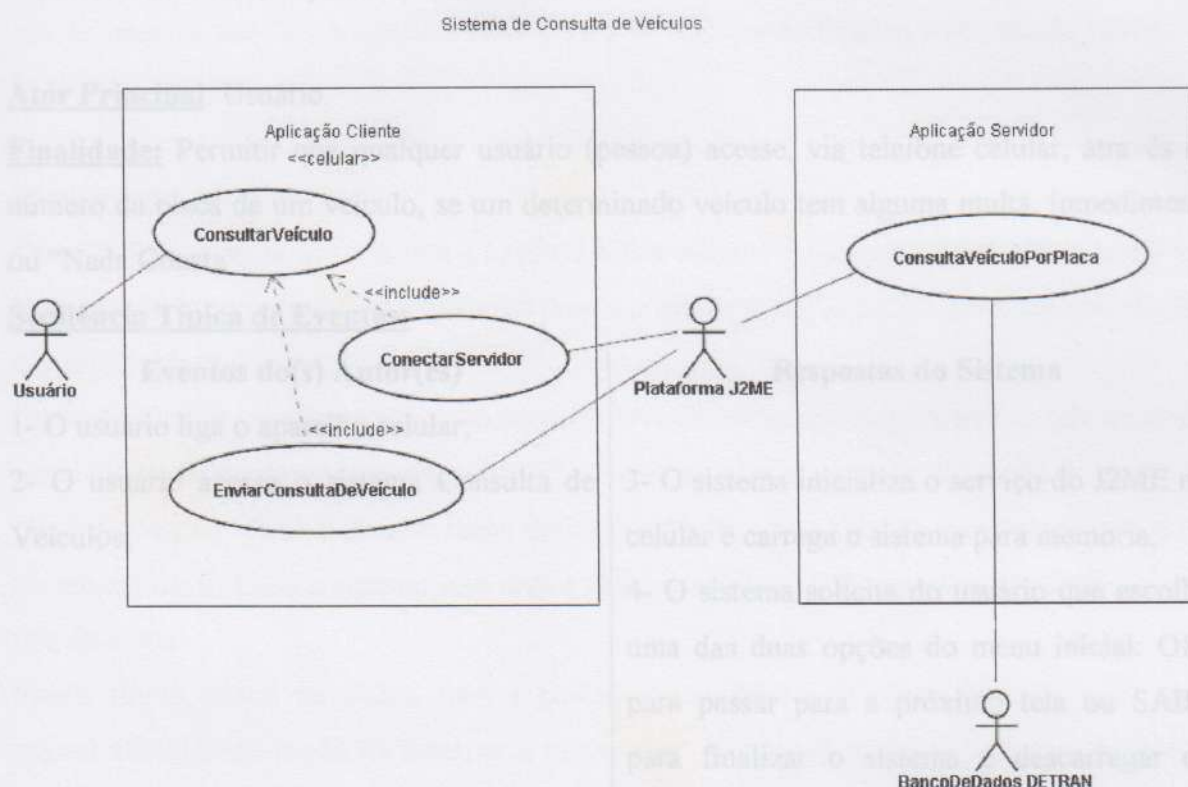


Figura 20 - Diagrama de Caso de Uso do Sistema

Fonte: O próprio autor

4.2.4.1 Caso de Uso de Alto Nível "ConsultarPlaca"

Casos de Uso:	Consultar placa
Atores:	Usuário, Plataforma J2ME, Banco de dados do DETRAN
Tipo:	Primário
Descrição:	O usuário liga o celular e acessa o sistema Consultar Veículo que irá solicitar do usuário a digitação do

número da placa do veículo, após selecionar no menu de opções e pressionar em OK, é feito uma conexão, via plataforma J2ME a um serviço Web, que irá fazer uma consulta a base de dados do DETRAN, passando como parâmetro o número da placa digitado.

Figura 21 - Caso de Uso Alto Nível

Fonte: O próprio autor

4.2.4.2 Caso de Uso Expandido “ConsultarPlaca”

Ator Principal: Usuário.

Finalidade: Permitir que qualquer usuário (pessoa) acesse, via telefone celular, através do número da placa de um veículo, se um determinado veículo tem alguma multa, impedimento ou “Nada Consta”.

Seqüência Típica de Eventos:

Eventos do(s) Autor(es)	Respostas do Sistema
1- O usuário liga o aparelho celular;	
2- O usuário acessa o sistema Consulta de Veículos;	3- O sistema inicializa o serviço do J2ME no celular e carrega o sistema para memória;
	4- O sistema solicita do usuário que escolha uma das duas opções do menu inicial: OK, para passar para a próximo tela ou SAIR, para finalizar o sistema e descarregar da memória;
5- O usuário irá selecionar a opção OK do menu do sistema;	6- O sistema solicita do usuário que digite um número de placa de veículo ou escolha uma das opções do menu: OK, SAIR, CANCELAR;
7- O usuário digita um número de placa de um veículo e seleciona OK;	8- O sistema faz a conexão com o Web Service passando como parâmetro o número da placa do veículo digitada;
	9- O Web Service conecta-se como o banco de dados do DETRAN, via JDBC, passando como parâmetro o número da placa do

4.2.5 Diagrama de Classes Proposto

O sistema será composto das classes classe *AplicaçãoConsulta* que é executada na plataforma J2ME e tem por objetivo pegar o

12- O usuário verifica a mensagem retornada e seleciona a opção SAIR para retornar a tela do menu inicial do sistema;

13- O usuário escolhe a opção SAIR para finalizar o sistema;

veículo;

10- O banco de dados retorna a mensagem para o WebService;

11- O Web Service retorna a mensagem para o Sistema, do celular;

14- O Sistema finaliza o serviço do J2ME .

Seqüência(s) Alternativas:

Passo 5: Caso o usuário selecione a opção SAIR o sistema finaliza sua operação e descarrega da memória, caso contrário o sistema irá para a próxima tela que pedirá que o usuário digite o número da placa do veículo.

Passo 6: Caso o usuário selecione a opção CANCELAR, o sistema irá para a tela de alerta, onde irá aparecer uma mensagem no celular e retornará para a tela de digitação de uma nova placa de veículo. Caso o usuário tenha digitado a opção sair o sistema irá retornar para a tela do menu inicial. Caso o usuário não digite nada e selecione OK, o sistema também irá para tela de alerta.

Passo 10: O banco de dados após a pesquisa do número da placa digitada, caso tenha encontrado alguma multa irá retornar a mensagem "Existe(m) multa(s) para este Veículo!", Caso haja algum impedimento irá retornar a mensagem "Existe(m) impedimento(s) para este Veículo!", caso existam multas e impedimento irá retornar a mensagem "Existe(m) multa(s) e/ou impedimento(s) para este Veículo!", caso contrário irá retornar a mensagem "Nada Consta para este Veículo!".

Passo 12: O usuário caso queira fazer nova consulta é só selecionar a opção CANCELAR no menu, caso contrário selecionar a opção SAIR para retornar para tela inicial.

4.2.5 Diagrama de Classes Proposto

O sistema será composto das classes representadas pela figura 22, logo abaixo. A classe *AppletConsulta* que é executada no dispositivo móvel, sendo implementada na plataforma J2ME e tem por objetivo pegar o número da placa do veículo digitado e repassar para o *Web Service* “WSDetran”, onde irá pegar a requisição feita e repassar a solicitação para a classe “Detran”. A classe “Detran” irá passar a solicitação para classe “Veiculo” que fará uma consulta, passando como parâmetro o número da placa do veículo, para verificar se existe alguma multa, em caso afirmativo, será retornado true (verdadeiro) para classe “Detran” que passará para classe “WSDetran”, esta por sua vez irá montar a string de retorno para que o dispositivo móvel possa mostrá-la na sua tela. Este mesmo processo será feito para o caso do veículo estar com algum impedimento, nesse caso o atributo “impedido” da classe “Veiculo” irá para “true” (verdadeiro), esse valor também é retornado para classe “Detran” para que esta possa enviar para classe “WSDetran” e montar a string de retorno para ser mostrado no dispositivo móvel. É bom salientar que não foram implementados todos os atributos e operações de todas as classes, foi mostrado apenas o que era de interesse para o objetivo desta monografia.

Figura 22 - Diagrama de Classes
Fonte: O próprio autor

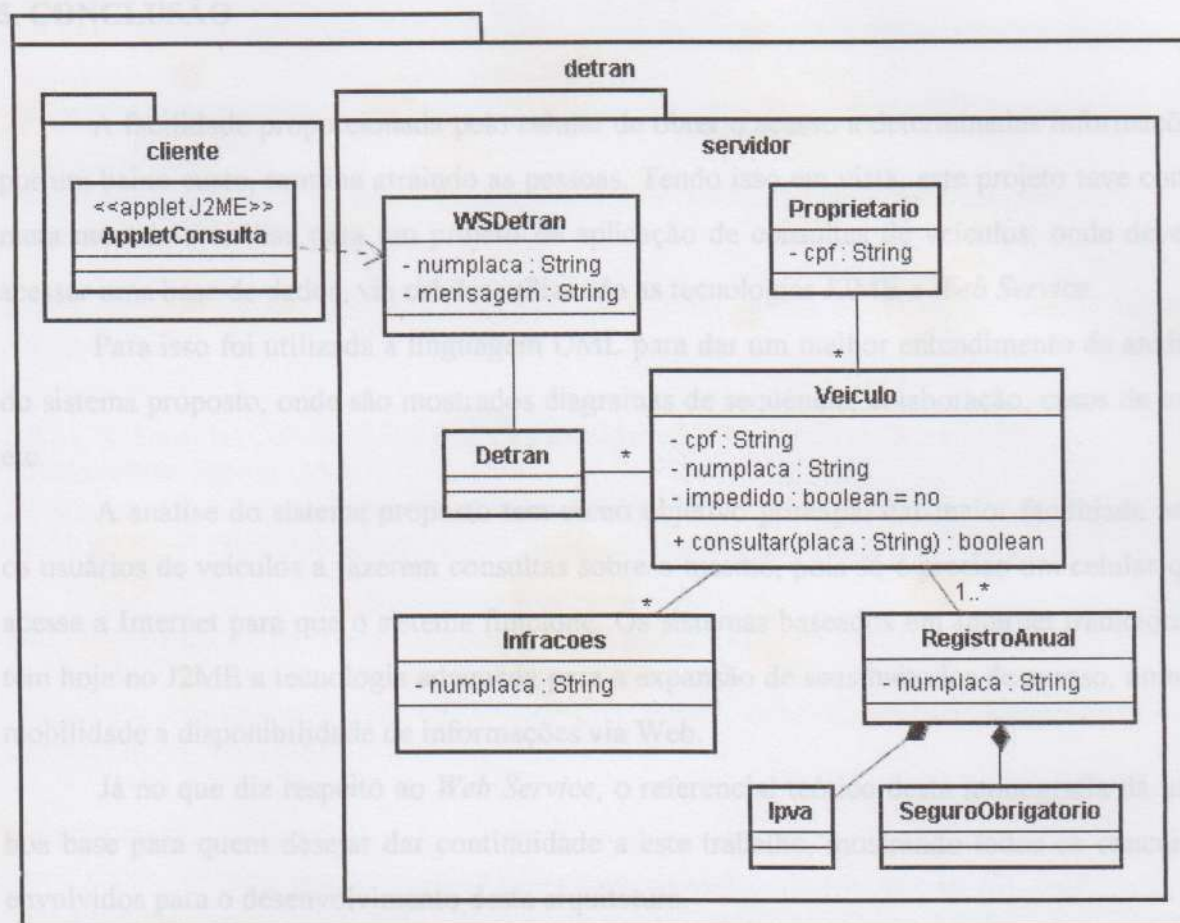


Figura 22 - Diagrama de Classes

Fonte: O próprio autor

5. CONCLUSÃO

A facilidade proporcionada pelo celular de obter o acesso a determinadas informações por um baixo custo, termina atraindo as pessoas. Tendo isso em vista, este projeto teve como meta mostrar a análise para um projeto de aplicação de consultas de veículos, onde deverá acessar uma base de dados, via celular utilizando as tecnologias J2ME e *Web Service*.

Para isso foi utilizada a linguagem UML para dar um melhor entendimento da análise do sistema proposto, onde são mostrados diagramas de seqüência, colaboração, casos de uso, etc.

A análise do sistema proposto tem como objetivo principal dar maior facilidade para os usuários de veículos a fazerem consultas sobre o mesmo, pois só é preciso um celular que acesse a Internet para que o sistema funcione. Os sistemas baseados em Internet tradicionais têm hoje no J2ME a tecnologia adequada para a expansão de seus métodos de acesso, unindo mobilidade à disponibilidade de informações via Web.

Já no que diz respeito ao *Web Service*, o referencial teórico desta monografia dá uma boa base para quem desejar dar continuidade a este trabalho, mostrando todos os conceitos envolvidos para o desenvolvimento desta arquitetura.

Como o foco desta monografia foi ser um referencial teórico de como desenvolver uma aplicação para plataforma J2ME, mostrando uma estrutura mínima. Pensando nisso, o item 4.1, desta monografia, mostra como criar um aplicativo bem simples, passo-a-passo onde é mostrada a mensagem "Exemplo de Aplicação para J2ME Alô Detran", na tela do celular. E, no item 4.2, traz uma análise de sistema, mostrando os diagramas de caso de uso com o diagrama de classes de como pesquisar um determinado veículo utilizando, para isso, o número da placa, do mesmo, digitado.

Com isso pode-se constatar que esta monografia atendeu seu objetivo maior que é de dar um bom embasamento a quem pretender desenvolver aplicativos utilizando a plataforma J2ME com *Web Service*.

DIAS, Kleia L.; FONTES, Wesley P. Desenvolvimento de Aplicações para Dispositivos Móveis Utilizando J2ME. Belém: Universidade Federal do Pará, 2003, 72p.

GOSLING, James. A Brief History of Green Project. 2003. Disponível em: <<http://java.sun.com/people/jag/green/index.html>>. Acesso em: 13 ago 2006.

HENDRIKS, Mark et al. Professional Java Web Services. Rio de Janeiro: Alta Books, 2002.

REFERÊNCIAS

- ANQUETIL, Nicolas. **Desenvolvimento de Software Orientado a Objetos**. 2003. Disponível em: <http://www.ucb.br/ucbtic/mgcti/paginapessoalprof/Nicolas/Disciplinas/UML/uml.pdf>. Acessado em: 25 out. 2006.
- BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML, Guia do Usuário**. Rio de Janeiro: Campus, 2000.
- BRAUN, Daniela. **Celular: 64% dos Consumidores Têm renda de R\$ 480,00 e Escolaridade Média**. 2006. Disponível em: http://computerworld.uol.com.br/comunicacoes/2006/11/29/idgnoticia.2006-11-29.4392147778/IDGNoticia_view. Acessado em: 06 dez. 2006.
- BRITO, Robinson Cris. **Desenvolvimento de Aplicativos para Celular – J2ME**. Centro Federal de Educação Tecnológica do Paraná. Pato Branco. 2004. Disponível em: <http://www.portaljava.com/home/modules.php?name=Content&pa=showpage&pid=76>. Acesso em: 19 maio 2006.
- CAMELO, Dioclécio Moreira. **Apostilha de Web Service**. 2002. Disponível em: <http://www.cqgp.sp.gov.br/downloads/WebServices.pdf>. Acesso em: 04 abr. 2006.
- CARNIEL, Juliano; TEIXEIRA, Clóvis. **Apostilha de J2ME**. 2003. Disponível em: <http://www.portaljava.com/home/modules.php?name=Content&pa=showpage&pid=24>. Acesso em: 19 mai. 2006.
- CONALLEN, Jim. **Desenvolvimento de Aplicações Web com UML**. 2.ed. Rio de Janeiro: Campus, 2003.
- CUNHA, Davi. **Web Services, SOAP e Aplicações Web**. 2002. Disponível em: http://devedge-temp.mozilla.org/viewsource/2002/soap-overview/index_pt_br.html. Acesso em: 15 mar. 2006.
- DEITEL, H. M.; Deitel, P. J. **Java Como Programar**. 3. ed. Porto Alegre: Bookman, 2001.
- DIAS, Klessis.L.; FONTES, Weskley P. **Desenvolvimento de Aplicações para Dispositivos Móveis Utilizando J2ME**. Belém: Universidade Federal do Pará; 2003, 72p.
- GOSLING, James. **A Brief History of Green Project**. 2003. Disponível em: <http://java.sun.com/people/jag/green/index.html>. Acesso em: 15 ago 2006.
- HENDRIKS, Mack et al. **Profissional Java Web Services**. Rio de Janeiro: Alta Books, 2002.

JEVEAUX, Paulo C. M. **Aprenda a Utilizar JDBC**. 2003. Disponível em:
 <<http://portaljava.com.br/home/modules.php?name=Content&pa=showpage&pid=5&page=3>.>
 >_Acesso em: 29 ago 2006.

LAUREANO, Jeziel. **Construção de um Componente para Geração de Código a partir de Diagramas UML**. 2002. Monografia (Curso Ciências da Computação) – Universidade Luterana do Brasil, Gravataí, 2002.

LARMAN, Craig. **Utilizando UML e Padrões**. 2. ed. Porto Alegre: Bookman, 2004.

MATOS, A.V. **UML: Prático e Descomplicado**. 2. ed. São Paulo: Érica, 2002.

MORISSEAU-LEROY, Nirva; SOLOMOM, M. K.; BASU, J. **Oracle8i: Programação de componentes Java com EJB, CORB e JSP**. Rio de Janeiro: Campus, 2001.

ORRICO, Ângelo C. A. **Desenvolvendo uma Aplicação J2ME**. 2006. Disponível em:
 <<http://www.devmedia.com.br/articles/viewcomp.asp?comp=2020>>. Acessado em: 15 ago. 2006.

OLIVEIRA, Talmai. **Dispositivos Móveis Portáteis, Aplicações Embarcadas. Plataforma J2ME**. Disponível em: <http://twiki.im.ufba.br/pub/Residencia/Cursos/curso_parte1.pdf>. Acessado em: 21 mar 2006.

PAMPLONA, Vitor F. **Web Services via J2SE e J2ME**. 2004. Disponível em:
 <http://www.imasters.com.br/artigo/2741/java/web_services_via_j2se_e_j2me>. Acesso em: 15 mar. 2006.

PAULAN, Renan. **Web Services in Java**. 2004. Disponível em:
 <<http://www.imasters.com.br/artigo/1863>>. Acesso em: 15 mar. 2006.

PEREIRA, Fernando M. Q. et al. Chamada Remota de Métodos na Plataforma J2ME/CLDC. **Revista Científica Periódica – Telecomunicações**, Minas Gerais, v.7, n.1, p.21-31, jul. 2004.

RIBEIRO, Nuno M. **A utilidade das tecnologias de computação móvel no ensino: o caso da gestão pessoal do professor**. 2004, 15p. Lisboa, Portugal: Universidade Fernando Pessoa. Disponível em: <http://www.cerem.ufp.pt/~nrbeiro/publicações/nrbeiro_socinf.pdf>. Acesso em: 13 ago. 2006.

SAMPAIO JÚNIOR, Antonio B. C.. **Java 2 Micro Edition II**. 2004. Disponível em:
 <<http://www.portaljava.com/home/modules.php?name=Content&pa=showpage&pid=68>>. Acessado em: 21 mai 2006.

SCHAEFER, Carine. **Prótipo de Aplicativo Para a Transmissão de Dados a Partir de Dispositivos Móveis Aplicados a uma empresa de Transporte**. 2004. Disponível em: <<http://www.inf.furb.br/~pericas/orientacoes/J2METransporte2004.pdf>> Acessado em: 21 mai 2006.

SOUZA, Vinícius Costa. **Minicurso Web Services com PHP**. 2004. Disponível em: <<http://www.inf.unisinos.br/~vinicius>>. Acesso em: 21 mar. 2006.

TODD, Nick; SZOLKOWSKI, Mark. **Java Server Pages – O Guia do Desenvolvedor**. Rio de Janeiro: Campus, 2003.