MELISSA FIORINI

UMA ARQUITETURA GENÉRICA DE SOFTWARE PARA DISPONIBILIZAÇÃO DE UMA APLICAÇÃO WEB PARA DISPOSITIVOS MÓVEIS

FLORIANÓPOLIS
2006

UNIVERSIDADE FEDERAL DE SANTA CATARINA

PROGRAMA DE PÓS-GRADUAÇÃO
EM ENGENHARIA ELÉTRICA

UMA ARQUITETURA GENÉRICA DE SOFTWARE PARA DISPONIBILIZAÇÃO DE UMA APLICAÇÃO WEB PARA DISPOSITIVOS MÓVEIS

Dissertação submetida à
Universidade Federal de Santa Catarina
como parte dos requisitos para a
obtenção do grau de Mestre em Engenharia Elétrica

MELISSA FIORINI

Florianópolis, Abril de 2006

UMA ARQUITETURA GENÉRICA DE SOFTWARE PARA DISPONIBILIZAÇÃO DE UMA APLICAÇÃO WEB PARA DISPOSITIVOS MÓVEIS

Melissa Fiorini

‘Esta Dissertação foi julgada adequada para obtenção do Título de Mestre em Engenharia Elétrica, Área de Concentração em Automação e Sistemas, e aprovada em sua forma final pelo Programa de Pós-Graduação em Engenharia Elétrica da Universidade Federal de Santa Catarina’

_____

Prof. Ricardo José Rabelo, Dr.
Orientador

_____

Prof. Alexandre Trofino Neto, Dr.
Coordenador do Programa de Pós-Graduação em Engenharia Elétrica

Banca Examinadora:

_____

Prof. Ricardo José Rabelo, Dr.
Presidente

_____

Prof. Carlos Barros Montez, Dr.

_____

Prof. Leandro Buss Becker, Dr.

_____

Prof. Mario Dantas, Ph.D.

Agradecimentos

Quero agradecer a todas as pessoas que de alguma forma contribuíram para o início, meio e fim dessa dissertação. A minha família e a todos os meus amigos que sempre me apoiaram e acreditaram na minha capacidade de concluir esse trabalho. Entre tantas pessoas, alguns nomes não podem deixar de serem citados e os meus sinceros agradecimentos são:

Ao Professor João Israel Bernardo do CEFET-PR, Unidade de Pato Branco, que me incentivou a iniciar o mestrado e me auxiliou em todo processo de inscrição e seleção no Programa de Pós-Graduação em Engenharia Elétrica da UFSC.

Ao pessoal do laboratório GSIGMA da UFSC, principalmente a Rui Jorge Tramontin Júnior e a Carlos Eduardo Gesser que me auxiliaram no desenvolvimento do protótipo e que me ensinaram fundamentos sem os quais não teria concluído esse trabalho.

Ao professor Ricardo José Rabelo que aceitou ser meu orientador, sempre apoiou as minhas decisões e me ajudou em várias situações difíceis durante todo o período do mestrado.

Ao meu namorado André Fernando Schiochet que sempre me incentivou a concluir o mestrado e me auxiliou na preparação de muitas aulas na tarefa de ser professora e mestranda ao mesmo tempo.

A minha mãe Maria Helena Guth e minha irmã Vanessa Fiorini que acompanharam de perto todas as minhas dificuldades e sempre me apoiaram.

Ao meu pai Elton Luiz Fiorini e meu irmão Nataniel Arthur Fiorini que, mesmo de longe, sempre me incentivaram a concluir o mestrado.

MUITO OBRIGADA a todos!!!

Resumo da Dissertação apresentada à UFSC como parte dos requisitos necessários para a obtenção do grau de Mestre em Engenharia Elétrica.

UMA ARQUITETURA GENÉRICA DE SOFTWARE PARA DISPONIBILIZAÇÃO DE UMA APLICAÇÃO WEB PARA DISPOSITIVOS MÓVEIS

Melissa Fiorini

Abril, 2006.

Orientador: Prof. Ricardo José Rabelo, Dr.
Área de Concentração: Automação e Sistemas.
Palavras-chave: redes sem fio, computação móvel, aplicação web, XSLT.
Número de Páginas: 108

RESUMO:Com o avanço das tecnologias de redes sem fio e dos dispositivos móveis, o uso da Computação Móvel está crescendo continuadamente, tanto na área dos negócios como na área pessoal. Com a mobilidade é possível ter acesso à informação desejada a qualquer hora, em qualquer lugar e de forma rápida.

Dentro deste contexto, o que pode ser observado no mercado atual é que a maioria das aplicações para a Computação Móvel não tem dado suporte aos vários tipos de dispositivos móveis existentes. Geralmente, a solução está atrelada a determinado tipo, protocolo de comunicação e sistema operacional do dispositivo em questão.

Neste trabalho é proposta uma arquitetura genérica de software para a disponibilização de uma aplicação web para dispositivos móveis. Essa arquitetura possibilita a uma aplicação web, do lado servidor, ser disponibilizada para a Computação Móvel, de forma a ser acessada por um cliente independente do tipo de dispositivo móvel, do sistema operacional (tanto dos servidores como dos dispositivos móveis), do protocolo de comunicação e da arquitetura de rede sem fio utilizada.

O principal componente dessa arquitetura é o Processador da Camada de Apresentação, que disponibiliza a informação de acordo com o dispositivo que a solicitou. Ele é baseado em XSLT (*Extensible Stylesheet Language Transformation),* que é a parte mais importante dos padrões XSL (*Extensible Stylesheet Language)* do W3C.

Com XSLT é possível transformar arquivos XML para qualquer linguagem de apresentação desejada, como HTML, WML, cHTML e XHTML. Isso é feito com a criação de *templates* específicos para cada linguagem de apresentação e para o tipo de dispositivo desejado, com a vantagem de que, além da transformação, é possível escolher quais são os elementos relevantes de serem visualizados de acordo com o tipo de dispositivo móvel e da aplicação em questão, basta que estes sejam especificados nos *templates.*

Abstract of Dissertation presented to UFSC as a partial fulfillment of the requirements of the degree of Master in Electrical Engineering.

A GENERIC ARCHITECTURE OF SOFTWARE FOR AVAILABILITY OF A WEB APPLICATION FOR MOBILE DEVICES

Melissa Fiorini

April, 2006.

Advisor: Ricardo José Rabelo, Dr.
Area of Concentration: Automation and Systems.
Keywords: wireless networks, mobile computing, web application, XSLT.
Number of Pages: 108

ABSTRACT: With the advance of the wireless technologies and of the mobile devices, the use of the Mobile Computing is growing continually, in such a way in the businesses area as in the personal area. With the mobility is possible to have access to the information at any time, at any place and in a fast way.

Inside this context what can be observed at the current market is that most of the applications for Mobile Computing has not been giving support to the several types of existing mobile devices. Usually the solution is harnessed a the certain type, communication protocol and operating systems of the device in subject.

In this work a generic architecture for availability of web appplication for mobile devices is proposed . This architecture makes possible to an application web, of the server side, to be available for the Mobile Computing, in way to be had accessed by a client independently of the type of mobile device, of the operational system (in such a way of the servers as of the mobile devices), of the protocol of communication and the wireless network used.

The main component of this architecture is the Processor of the Presentation Layer that avails the information according to the device that requested it. It is based on XSLT (Extensible Stylesheet Language Transformation) the most important part of standards XSL (Extensible Stylesheet Language) of the W3C.

With XSLT is possible to transform archives XML for any other presentation language, either HTML, or WML, or HTML, or XHTML.This is made with the creation of specific templates for each presentation language and the type of desired device, with the advantage of that, beyond the transformation, it is possible to choose which are the excellent elements to be visualized in accordance with the type of mobile device and the application in question, it is enough that these are specified in templates.

Sumário

Lista de Figuras

1 Introdução

1.1 Cenário Geral

Hoje em dia, com a globalização da economia e o mercado competitivo, as empresas estão investindo cada vez mais em tecnologia da informação (TI) para ampliarem seus negócios. A Internet já é uma ferramenta consagrada para a troca de informações e para transações comerciais que facilita e simplifica o acesso à informação para clientes, funcionários, parceiros, investidores, acionistas e fornecedores.

Devido ao avanço das tecnologias de redes sem fio e dos dispositivos móveis, o uso da Computação Móvel está crescendo continuamente, tanto na área dos negócios, como na área pessoal. A Computação Móvel é um paradigma da computação no qual usuários portando dispositivos móveis têm acesso a serviços via redes sem fio independente de sua localização (FORMAN e ZAHORJAN, 1994). Cada vez mais executivos estão trocando *notebooks* por telefones inteligentes, que buscam e exibem *on-line*, todas as informações importantes para a tomada de decisões longe do escritório (TEIXEIRA JUNIOR, 2005). Com a mobilidade é possível ter acesso à informação desejada a qualquer hora, em qualquer lugar e de forma rápida.

A cada dia surgem no mercado novos dispositivos móveis; são *pagers*, telefones celulares, *smartphones*, PDAs (*Personal Digital Assistant), Handhelds, Palms*, etc., e cada um tem características diferentes, com suas restrições de tamanho de tela, teclado, poder de processamento, protocolo de comunicação, sistema operacional e serviços fornecidos. O desenvolvimento de uma aplicação para a Computação Móvel deve levar em consideração esses fatores.

Quando se trata do desenvolvimento de uma aplicação web para Computação Móvel vários tipos de aplicações podem ser citados, como *e-mails*, *chats*, *games*, compras *on-line*, notícias, etc. E para caracterizar o tipo de aplicação a ser desenvolvido, não se pode deixar de falar de uma das tendências de maior impacto nos processos de negócios das empresas: a área de *Mobile Business* ou *m-business*.

Antes de definir *m-business*, é necessário que *e-business* seja definido, pois *m-business* nada mais é do que uma extensão do mesmo. Definindo, *e-business* é a integração dos sistemas, dos processos, das organizações, das cadeias de valor e de mercados

utilizando tecnologias e conceitos baseados na Internet. E, por analogia, *Mobile Business* ou *m-business* pode ser definido como uma coleção de tecnologias móveis e aplicações utilizadas para dar suporte a processos, cadeias de valor e mercados utilizando redes sem fio (STANOEVSKA-SLABEVA, 2003).

Outra definição para *m-business* inclui todas as atividades relacionadas a transações comerciais em potencial através de redes de comunicação que se conectam com dispositivos móveis (PIGNEUR e CAMPONOVO, 2003). Pode-se dizer ainda que *M-business* é a infra-estrutura de aplicação necessária para manter negócios e para vender informação, produtos e serviços utilizando dispositivos móveis (ROBINSON e KALAKOTA, 2001).

No entanto, para tornar as aplicações web disponíveis também para os dispositivos móveis, segundo o modelo *m-business*, é necessária uma adaptação no conteúdo para que ele possa ser acessado por equipamentos com telas e teclados menores e com memória e poder de processamento limitados. Portanto, os desenvolvedores de aplicações web precisam repensar a navegação para que seus usuários possam acessar as informações com maior facilidade (MELLO e REIS, 2005).

Outro ponto importante de uma aplicação para *m-business* é a escolha da plataforma de desenvolvimento. Uma plataforma de desenvolvimento para *m-business* deve suportar padrões abertos e uma variedade de dispositivos, deve ser independente de hardware, das redes, dos protocolos e dos dispositivos. Isto permite deixar a escolha dos dispositivos a cargo do cliente.

Construir uma ponte ligando a mobilidade, a web, e os canais de comunicação tradicionais requer um planejamento cuidadoso para definir as experiências do cliente, os processos de negócio, os conteúdos, a infra-estrutura da aplicação e os modelos de gerenciamento. As empresas que não seguirem esta estratégia e não adotarem a mobilidade estarão perdendo em competitividade (ROBINSON *et al*., 2001).

Quanto à tecnologia *wireless* utilizada, questões como largura de banda, conexões, latência, segurança e criptografia devem ser tratadas igualmente de maneira transparente ao usuário do serviço acessado através da rede sem fio (BRANS, 2002).

O que se observa atualmente é a existência de uma boa infra-estrutura para comunicações móveis, um avanço tecnológico de software e hardware nos dispositivos móveis, e uma crescente necessidade das empresas disponibilizarem serviços e conteúdo

como forma de, tanto atenderem as necessidades do mercado e também melhorarem suas curvas de retorno de investimento.

Porém, quando se fala em disponibilização de conteúdo, é importante verificar no desenvolvimento da aplicação que tipo de informação é solicitado. Como por exemplo, na Figura 1, onde são mostradas comparações feitas por STANOEVSKA-SLABEVA (2003), que divide a Internet considerando o modo fixo e o modo móvel. Pelo que pode ser visto, no modo Móvel o cliente quer acesso rápido, simples e com conteúdo especializado, diferente do modo fixo, onde o cliente tem tempo de buscar a melhor opção dentre os vários serviços disponíveis.

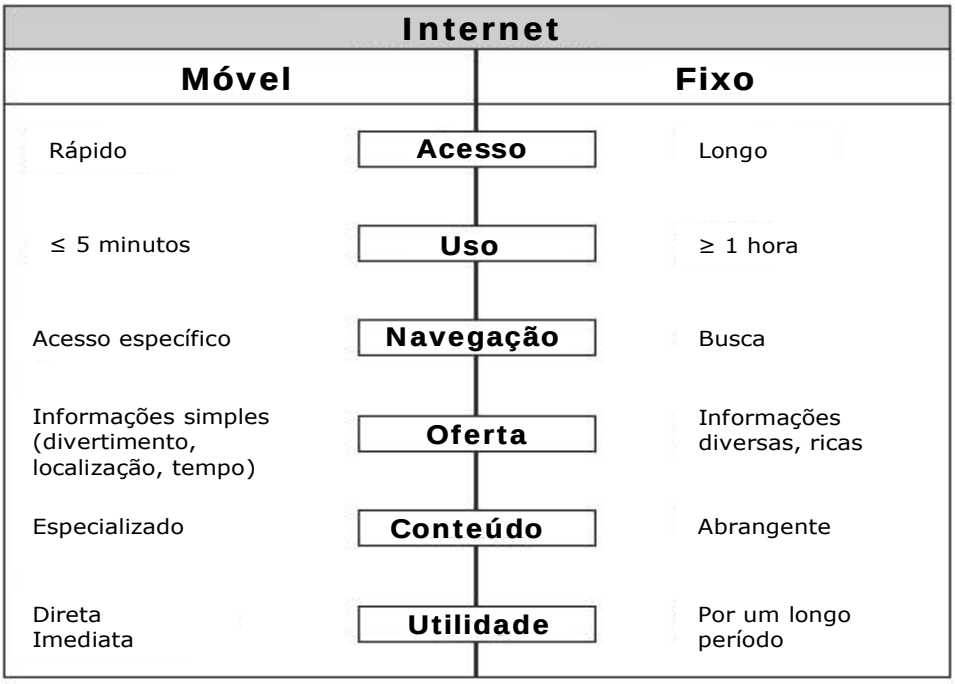

Figura 1: Comparações entre a Internet no Modo Fixo e no Modo Móvel. Adaptada de STANOEVSKA-SLABEVA, 2003.

Dentro deste contexto, o que pode ser observado no mercado atual é que a maioria das aplicações desenvolvidas para a Computação Móvel atuais não tem dado suporte aos vários tipos de dispositivos móveis existentes. Geralmente a solução fornecida está atrelada à determinada marca, tipo, protocolo de comunicação e sistema operacional do dispositivo em questão. Pode-se citar como exemplo o banco Bradesco, que disponibiliza serviços bancários apenas para PDAs que possuem sistema operacional Pocket PC ou Windows CE (BRADESCO, 2005). Existem empresas que fornecem soluções que possibilitam

disponibilizar o acesso a aplicações web a partir de vários tipos de dispositivos móveis, como no caso da IBM com o *WebSphere Transcoding Publisher* (IBM, 2005); porém são soluções proprietárias onde o cliente adquire a solução completa e a implanta na empresa, e nem sempre todo potencial da solução é utilizado dependendo das necessidades do mesmo, pois uma solução deste tipo é desenvolvida pensando-se nos vários tipos de clientes que possivelmente irão adquiri-la e o pacote é fechado e vendido aos clientes sem distinção.

Portanto, ao prescindirem de soluções mais genéricas, há um aumento de custo e tempo de desenvolvimento quando se deseja ampliar a gama de dispositivos móveis para os quais se deseja disponibilizar conteúdo e serviços e um conseqüente maior tempo de resposta ao mercado.

1.2 Objetivo

O objetivo geral deste trabalho é propor uma arquitetura genérica de software, baseada em padrões, que possibilite a uma aplicação web, do lado servidor, ser disponibilizada para a Computação Móvel de forma a poder ser acessada por um cliente, independente do tipo de dispositivo móvel, do sistema operacional (tanto dos servidores como dos dispositivos), do protocolo de comunicação e da arquitetura de rede sem fio que vai ser utilizada.

O trabalho proposto compreende apenas a disponibilização de um dado conteúdo sem englobar questões associadas a eventuais adaptações dinâmicas, tais como adaptação de acordo com o contexto, localização geográfica, etc.

1.2.1 Objetivos Específicos

- Estudo das tecnologias de redes sem fio, avaliando suas aplicabilidades, vantagens e limitações;
- Concepção e implementação de uma arquitetura genérica de software para que aplicações web, instaladas no lado servidor, possam ser acessadas por quaisquer dispositivos móveis;
- Validação da arquitetura e implementação num sistema de gestão de empresas virtuais, permitindo não apenas o acesso à informação como também a entrada de dados;

- Criação de um guia de recomendações para a implantação da arquitetura proposta em sistemas legados e em sistemas a serem desenvolvidos;
- Disponibilização do módulo implementado na Internet.

1.3 Organização da Dissertação

O capítulo 1 enquadrou o problema e motivou para a necessidade de soluções mais genéricas e abertas para o acesso a aplicações web por intermédio de dispositivos móveis.

O Capítulo 2 apresenta as principais tecnologias de redes sem fio e suas aplicações, e uma avaliação geral dessas tecnologias frente ao objetivo proposto no trabalho.

O Capítulo 3 apresenta tecnologias e padrões de desenvolvimento Web utilizados em trabalhos relacionados. Alguns destes serão aplicados na implementação do protótipo de software proposto.

O Capítulo 4 apresenta os trabalhos relacionados, mostrando soluções proprietárias, aplicações, trabalhos acadêmicos e projetos na área de redes sem fio onde o principal objetivo é disponibilização de aplicações web para dispositivos móveis.

O Capítulo 5 apresenta o modelo proposto na forma de uma arquitetura genérica de software para a disponibilização de uma aplicação web para dispositivos móveis.

O Capítulo 6 apresenta os aspectos de implementação do módulo proposto na arquitetura a partir do desenvolvimento do protótipo de software.

Finalmente, o Capítulo 7 apresenta as conclusões sobre os resultados obtidos com o presente trabalho e algumas propostas para trabalhos futuros.

2 Redes Sem Fio

Neste capítulo serão apresentadas as principais tecnologias de redes sem fio partindo de arquiteturas de redes e protocolos genéricos para as plataformas de comunicação existentes e suas aplicações, finalizando com uma avaliação geral dessas tecnologias frente ao objetivo proposto no trabalho.

2.1 Arquiteturas de Rede e Protocolos Genéricos para Redes Sem Fio

2.1.1 Redes Infra-estrutura

Redes infra-estrutura, como mostra a Figura 2, são redes sem fio onde redes cabeadas são utilizadas como um *backbone* que conecta as Estações-base.

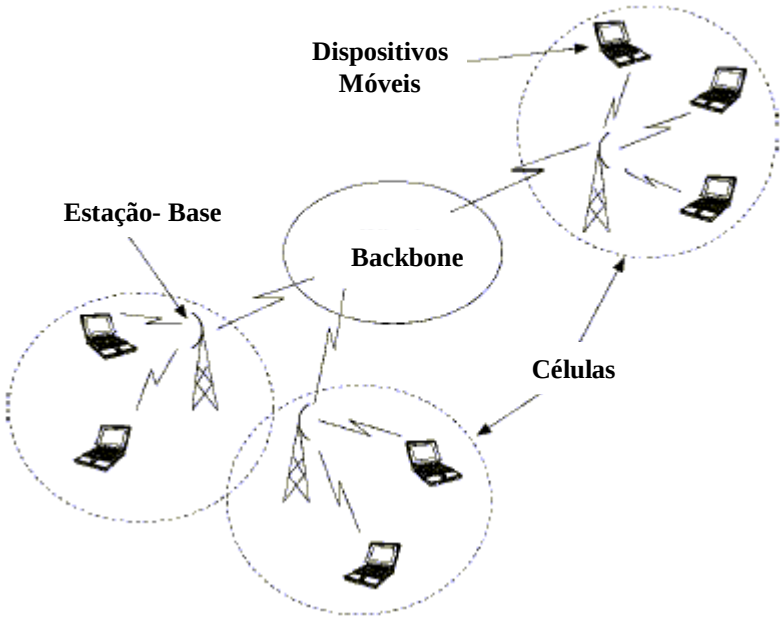

Figura 2: Rede Infra-estrutura. Traduzida de SIVALINGAM *et al*., 2001.

As Estações-base são computadores convencionais e estações de trabalho equipadas com cartões adaptadores para redes sem fio. Elas são responsáveis pelo controle de acesso para um ou mais canais de transmissão para dispositivos móveis localizados dentro da área de cobertura (SIVALINGAM *et al*., 2001).

O sinal transmitido pelos canais de transmissão pode ser em: FDMA (*Frequency Division Multiple Access*) onde a faixa de freqüências disponível é subdividida em bandas parciais, as quais são atribuídas aos vários usuários, TDMA (*Time Division Multiple Access*) que consiste na divisão de cada canal de freqüência em períodos de tempo, assim cada usuário ocupa um espaço de tempo específico na transmissão, ou CDMA (*Code Division Multiple Access*) onde todos usuários transmitem seus sinais ao mesmo tempo e nas mesmas freqüências e cada um possui um código binário exclusivo para diferenciá-lo.

2.1.2 Redes *Ad Hoc*

Por outro lado, redes *Ad hoc* são redes sem fio nas quais um conjunto de dispositivos coopera para manter a conectividade da rede. Esta arquitetura de rede é completamente desprovida de fios. Todos os dispositivos dentro da área de cobertura se comunicam ponto-a-ponto, envolvendo duas unidades**,** sem envolver pontos de acesso centrais. Um exemplo da topologia *Ad hoc* é mostrado na Figura 3.

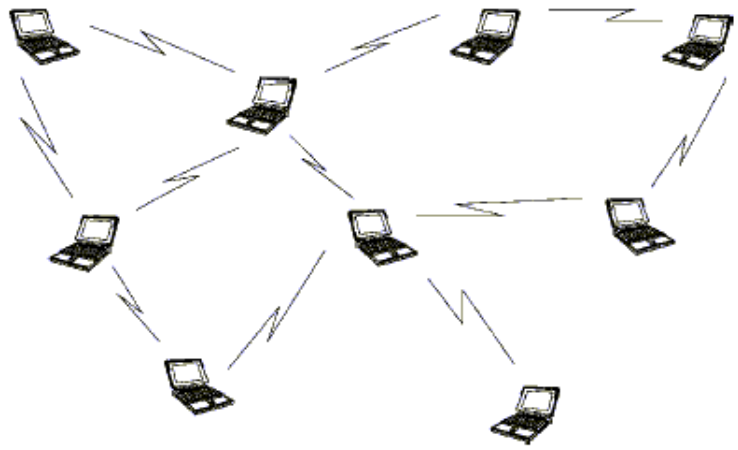

Figura 3 : Rede *Ad hoc* (SIVALINGAM *et al*., 2001).

Os dispositivos de uma rede *Ad hoc* podem se mover arbitrariamente. Deste modo, a topologia da rede muda freqüentemente e de forma imprevisível. Assim, a conectividade entre os dispositivos móveis muda constantemente, requerendo uma permanente adaptação e reconfiguração de rotas.

Redes *Ad hoc* são úteis em situações nas quais a conexão é necessária temporariamente. Exemplos de uso são ambientes militares e coberturas de desastres.

2.1.3 Protocolos X Camadas

Nesta seção é dada uma introdução ao software utilizado em sistemas de redes de dados sem fio. Programas de aplicação utilizando a rede não interagem diretamente com o hardware de rede. Ao contrário, uma aplicação interage com o software do protocolo. A noção de camadas de protocolos provê uma base conceitual para o entendimento do conjunto de protocolos que junto com o hardware formam o sistema de comunicação (SIVALINGAM *et al*., 2001).

A pilha de protocolos OSI (*Open Systems Interconnection*) tradicional é utilizada para mostrar, conceitualmente, as funções de cada camada de um sistema de comunicação sem fio genérico e está mostrada na Figura 4.

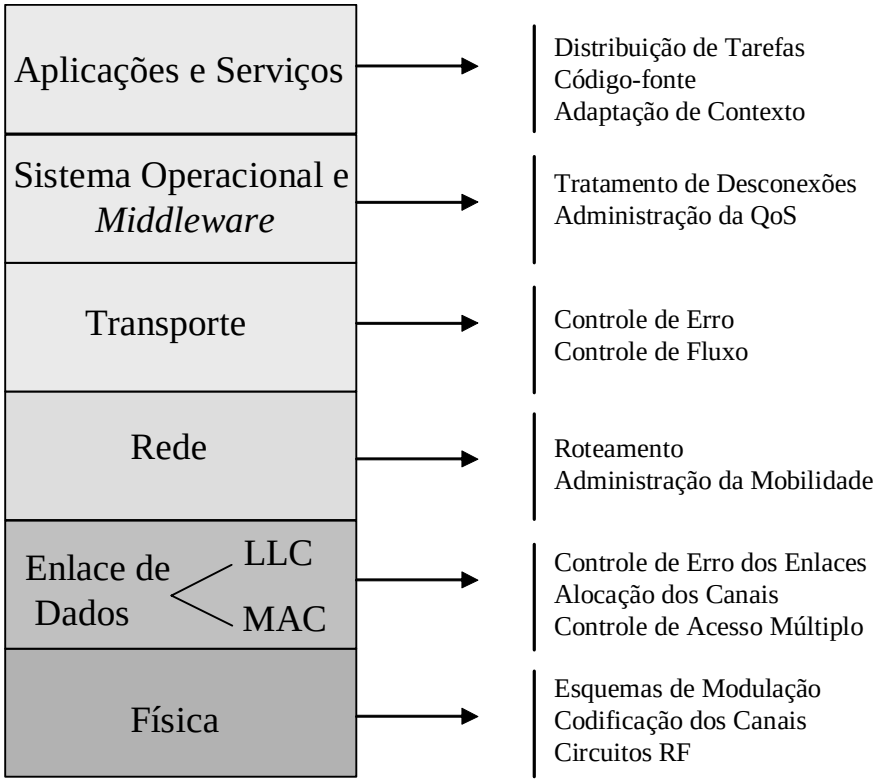

Figura 4: Pilha de Protocolos de uma Rede sem Fio Genérica. Traduzida de SIVALINGAM *et al*., 2001.

A camada de aplicação e serviços ocupa o topo da pilha seguida da camada de sistema operacional e *middleware*, transporte, rede, enlace e física que são descritas a seguir:

- Camada Física - A camada física consiste dos circuitos de RF (radiofreqüência), modulação e sistemas de codificação do canal.
- Camada de Enlace de Dados - A camada de enlace é responsável pelo estabelecimento de conexões e procedimentos que permitem o uso eficiente dos meios de transmissão. Uma subcamada da camada de enlace, a LLC (*Logical Link Control*) é responsável pelo controle de erros no enlace sem fio, segurança e retransmissão de pacotes. Outra subcamada da camada de enlace, a MAC (*Media Access Control*) é responsável por coordenar o acesso ao meio entre dispositivos compartilhando canais sem fio em uma região.
- Camada de Rede - A camada de rede é responsável pelo roteamento de pacotes, estabelecendo o tipo de serviço de rede (conexão sem fio versus conexão orientada), e transferindo pacotes entre as camadas de transporte e enlace. Em um ambiente móvel esta camada também tem a responsabilidade de re-roteamento de pacotes e administração da mobilidade.
- Camada de Transporte - A camada de transporte é responsável pelo transporte dos dados fim-a-fim independente da camada física em uso.
- Camada de Sistema Operacional e *Middleware* - A camada de Sistema Operacional e *Middleware* administra o acesso a recursos físicos, como CPU, memória e espaço em disco de aplicações executadas no dispositivo, trata desconexões e administra a QoS (*Quality of Service*) dentro de dispositivos móveis. As tarefas convencionais são escalonamento de processos e administração do sistema de arquivos.
- Camada de Aplicação - A camada de aplicação é responsável pela distribuição de tarefas entre as estações fixas e móveis, pela codificação e decodificação de áudio e vídeo e pela adaptação de contexto em um ambiente móvel. Os serviços desta camada são variados de acordo com a aplicação específica.

Os problemas inerentes ao meio de transmissão e questões relacionadas à mobilidade são desafios para o desenvolvimento de protocolos adequados para redes sem fio. Adicionalmente, a eficiência em termos de consumo de energia é primordial quando se trata de dispositivos portáteis com baixo poder de armazenamento.

2.2 Plataformas de Comunicação sem Fio para WWANs

Nas plataformas WWANs (*Wireless Wide Area Network*), utilizadas para transmissão de dados de sistemas celulares, a tecnologia usada na transmissão passou por uma evolução bastante significativa. A primeira geração ficou caracterizada pelo uso da tecnologia analógica para a transmissão de dados, evoluindo para a digital na segunda geração.

Antes mesmo de ser explorado todo o potencial da segunda geração, deu-se início ao desenvolvimento da terceira geração de sistemas móveis, com o objetivo principal de se obter um único padrão mundial nos sistemas celulares, além de oferecer aos usuários taxas de transmissão mais altas com melhor qualidade.

2.2.1 Primeira Geração – 1G

A mobilidade para os usuários de telefonia veio no final dos anos 70 e início dos anos 80 quando os telefones celulares apareceram no mercado utilizando redes de telefonia analógicas centradas em voz.

O primeiro sistema telefônico celular foi o AMPS (*Advanced Mobile Phone Service*) desenvolvido pelos Laboratórios Bell da AT&T e lançado no início dos anos 80 em Chicago.

O AMPS é uma rede analógica de telefonia e, por utilizar sinal analógico, se adapta melhor para o transporte de voz do que de dados. Outras redes analógicas de telefonia foram colocadas no mercado, como NMT (*Nordic Mobile Telephone*) na Escandinávia, e TACS (*Total Access Communication System*), usada na China e em outros países (DUBENDORF, 2003).

Estas redes utilizam o conceito de células, onde o espaço geográfico é dividido em setores, cada um chamado de célula, para otimizar e reutilizar freqüências para poder ter um maior número de usuários. Uma região é dividida em diversas regiões hexagonais com a mesma freqüência sendo alocada para células não adjacentes, tornando possível o reuso das bandas de freqüência, conforme a Figura 5.

Todas estas redes, baseadas em tecnologias de modulação analógica, se caracterizam pela baixa capacidade espectral e facilidade na interceptação das conversações.

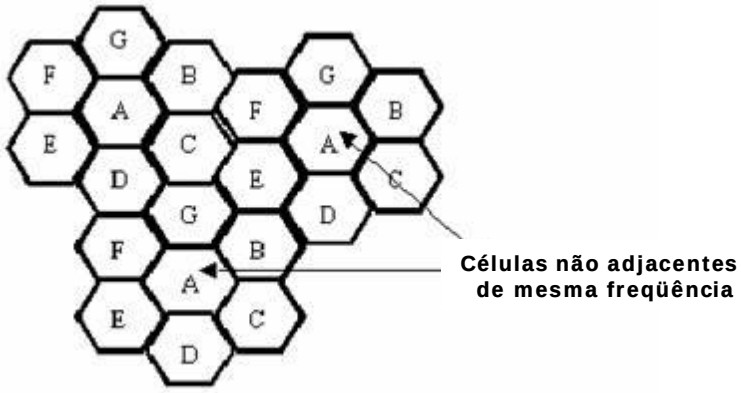

Figura 5: Conceito de reuso de freqüências.

2.2.2 Segunda Geração – 2G

A falta de padronização nos sistemas móveis utilizados na Europa, a falta de padrões internacionais, que afetam diretamente a indústria, e o limite da capacidade dos sistemas móveis de primeira geração nas maiores áreas metropolitanas dos EUA motivaram o desenvolvimento de sistemas digitais, surgindo a segunda geração de sistemas móveis.

Estes sistemas possuem, em princípio, além de maior capacidade, algumas vantagens em relação aos sistemas analógicos, como o uso de técnicas de codificação digital de voz mais poderosa, maior segurança nas informações, códigos de detecção e eliminação de erros.

A segunda geração de sistemas móveis é representada por sistemas como: o GSM (*Global System for Mobile Communications*) na Europa, o TDMA e o CDMA nos EUA, e o PDC (*Japanese Personal Digital Cellular*) no Japão. Todos estes sistemas possuem como característica comum o fato de usarem a tecnologia digital.

No TDMA é utilizada a técnica de multiplexação por divisão de tempo onde cada canal de voz é dividido em períodos de tempo. Isso possibilita até três conversações simultâneas no mesmo canal e assim é possível disponibilizar o mesmo canal para diversos usuários ao mesmo tempo (DUBENDORF, 2003).

O CDMA faz o uso da tecnologia de espectro espalhado. Ao invés de subdividir as bandas em unidades cada vez menores, o CDMA permite que todos os usuários dividam a mesma freqüência ao mesmo tempo.

O segredo de gerenciar esta tecnologia em que todos têm acesso à mesma faixa de freqüência está no empacotamento dos dados. Sinais de diferentes usuários são diferenciados por seqüências únicas de código. Cada comunicação de dados ou voz é

quebrada em pequenas partes, e para cada peça é dado um código identificador. No lado de recebimento dos pacotes, o conhecimento da seqüência de código que está sendo enviada permite que o sinal seja extraído e reconstituído (ROSENBERG e KEMP, 2002).

As redes GSM são as mais populares em todo o mundo, tendo uma grande base de clientes na Europa e na Ásia e possuindo mais da metade dos clientes mundiais.

GSM é baseado na tecnologia TDMA. Uma nova estrutura foi adotada nos terminais, que são compostos de um transceptor e um cartão SIM (*Subscriber Identity Module*). Este último é um pequeno *chip* que tem a função de armazenar todos os dados do usuário, a sua agenda pessoal e o seu código de autenticação. O cartão SIM pode ser desconectado de um transceptor e conectado a outro transceptor pelo próprio usuário, podendo este usá-lo em qualquer outro celular GSM, como se fosse o dele, pois todos os seus dados estão armazenados no *chip*. É o cartão SIM que dá à unidade móvel a sua identidade.

Tendo um elemento de identidade móvel, o usuário pode viajar para qualquer lugar do mundo onde as faixas de freqüência de operação do GSM são diferentes e continuar fazendo e recebendo chamadas, basta trocar de aparelho (WEBB, 1999).

2.2.3 Terceira Geração – 3G

Apesar de os sistemas de segunda geração não terem sido totalmente explorados ainda, já se trabalha intensamente no desenvolvimento da terceira geração, ou 3G. O ITU *(International Telecommunication Union*) elaborou um conjunto de requisitos para apresentar uma proposta para a RTT (*Radio Transmission Technology*). Tal proposta seria candidata a compor um conjunto de especificação da 3G (GUIMARÃES, 2001).

Os sistemas de terceira geração foram inicialmente chamados de FPLMTS (*Future Public Land Mobile Telecommunication System*), e tinham o objetivo de atender tanto aos usuários fixos quanto aos móveis, em redes públicas ou privadas. Por ser uma sigla de difícil pronúncia, o FPLMTS foi substituído pelo nome de IMT-2000 (*International Mobile Telecommunications),* onde o 2000 indica o ano em que os serviços foram inicialmente oferecidos (GUIMARÃES, 2001).

Nesse período, na Europa, o ETSI (*European Telecommunications Standards Institute*) iniciou o desenvolvimento de um sistema 3G com o objetivo de prover um padrão universal para as comunicações pessoais, com a qualidade de serviços equivalente à rede fixa. Tal sistema foi denominado de UMTS (*Universal Mobile Telecommunications*

System), que será a versão européia do IMT-2000. O IMT-2000 e o UMTS são padrões compatíveis e possuem capacidade para *roaming* em escala mundial.

O ITU prevê que as tecnologias dominantes para os sistemas 3G serão baseadas em sistemas CDMA. Isso ocorre porque os sistemas CDMA permitem alta flexibilidade para transmissão de altas taxas de dados e utilização de sinais recebidos por múltiplos percursos, resultando em um ganho na recepção de sinais. E como os usuários transmitem ao mesmo tempo e na mesma freqüência, um terminal móvel pode se comunicar com várias estações rádio-base ao mesmo tempo (GUIMARÃES, 2001).

Os sistemas 3G UMTS são baseados na tecnologia W-CDMA (*Wideband Code Division Multiple Access*), estando em operação na Europa e no Japão. Uma das chaves para o sucesso do W-CDMA é a sua integração com as atuais redes 2G, sejam elas baseadas na tecnologia GSM, TDMA ou CDMA.

Os novos sistemas UMTS (W-CDMA) são baseados em uma evolução do GSM, e, portanto, grande parte dos investimentos realizados em redes 2G não será perdida, pois vários elementos da rede poderão ser mantidos para o suporte do sistema UMTS.

Os sistemas IMT-2000 são baseados nas tecnologias W-CDMA e CDMA2000 (versão atualizada da norma CDMA). O W-CDMA é, pelo mesmo motivo, utilizado no sistema UMTS, comentado anteriormente. A tecnologia CDMA2000 possui a mesma característica que a CDMA; a diferença é que o CDMA2000 permite uma capacidade de voz adicional.

2.3 Plataformas de Comunicação Sem Fio para WLANs

2.3.1 Padrão 802.11

Lançado em 1997, depois de quase sete anos de desenvolvimento, o padrão oficial IEEE 802.11 tem o objetivo de desenvolver especificações para conectividade sem fio de estações fixas, portáteis e móveis dentro de uma área local (GEIER, 2001). As especificações desenvolvidas são para a camada física e camada MAC em WLANs (*Wireless Local Area Network*).

O padrão IEEE 802.11 define as arquiteturas de rede no modo infra-estrutura , com ao menos um ponto de acesso central conectado a uma rede cabeada e o modo *ad hoc* em

que os dispositivos se comunicam diretamente uns com os outros sem necessitar de um ponto de acesso central ou uma conexão de rede cabeada (VARSHNEY, 2003).

A Figura 6 mostra o padrão 802.11 e suas extensões juntamente com as principais características quanto ao espectro de freqüência utilizado, a taxa de transmissão e vantagens e desvantagens de cada extensão do padrão.

Padrão	Espectro	Taxa de transmissão	Maior Desvantagem	Maior Vantagem(s)
802.11	2.4 GHz	2 Mbps	Taxa de transmissão baixa	Alcance elevado
802.11a	5 GHz	54 Mbps	Menor alcance de todos padrões 802.11	Alta taxa de transmissão
802.11b	2.4 GHz	11 Mbps	Taxa de transmissão muito baixa para as novas aplicações	Extensamente desenvolvido Alcance elevado
802.11g	2.4 GHz	54 Mbps	Número limitado de WLANs	Alta taxa de transmissão em 2.4GHz

Figura 6: Características dos Padrões 802.11. Traduzida e adaptada de VARSHNEY, 2003.

Os padrões 802.11 e 802.11b apesar de possuírem um alcance elevado, em torno de 75 metros, possuem uma taxa de transmissão baixa na freqüência de 2.4GHz. Um ponto negativo nestes padrões é a alta interferência tanto na transmissão como na recepção de sinais, porque funcionam na freqüência de 2.4 GHz, denominada banda ISM (*Industrial, Scientific and Medical)*,equivalente aos telefones sem fio, fornos microondas e dispositivos *Bluetooth*.

Já o padrão 802.11a tem o menor alcance de todos os padrões, em torno de 25 metros, porém tem alta taxa de transmissão na freqüência de 5GHz. Entre as suas vantagens estão a velocidade, a gratuidade da freqüência utilizada e a ausência de interferências. Uma desvantagem é a incompatibilidade com os padrões 802.11, 801.11b e 802.11g.

O padrão 802.11g tem alta taxa de transmissão na freqüência de 2.4 GHz, tem um alcance em torno de 25 metros e é compatível com os padrões 802.11 e 802.11b, porém ainda há poucas WLANs que utilizam o padrão.

2.3.1.1 Futuro do Padrão 802.11

O futuro de WLANs é promissor, mas muitos obstáculos tecnológicos e relacionados a negócios ainda existem. Para que o desenvolvimento ocorra, os fabricantes e os

provedores de acesso de redes sem fio devem conhecer as demandas do consumidor para facilitar o acesso, melhorar a qualidade de serviço, ter mais ferramentas de gerência da rede e ter flexibilidade de preços - por exemplo, a opção de cobrança por transação é melhor que pela duração de uma conexão, como é atualmente o caso.

A taxa de transmissão também deve ser aumentada para aplicações de alta capacidade como multimídia e videoconferência em WLANs. Para resolver este problema, o IEEE 802 *High Throughput Task Group* criou o padrão 802.11n para aumentar a taxa de transferência para 108 Mbps e, possivelmente para 320 Mbps (VARSHNEY, 2003).

2.4 Plataformas de Comunicação Sem Fio para WPANs

2.4.1 Bluetooth

O *Bluetooth* SIG (*Bluetooth Special Interest Group*), formado em fevereiro de 1998 pelos líderes em comunicação e computação móvel Ericsson, IBM, Intel, Nokia, e Toshiba, desenvolve especificações para o *Bluetooth* também conhecido como padrão 802.15.1 para WPANs (*Wireless Personal Area Network*).

O *Bluetooth* é um padrão para conectividade sem fio entre celulares, *notebooks*, PDAs, e outros periféricos habilitando voz e dados via enlaces de rádio de curto alcance, permitindo aos usuários conexões com vários dispositivos fácil e rapidamente, sem necessidade de cabos.

2.4.1.1 A Arquitetura *Bluetooth*

O *Bluetooth* tem sido especificado e desenvolvido enfatizando-se robustez e baixo custo. Sua implementação está baseada sobre um alto desempenho, baixo custo e um transceptor de rádio integrado. Os alvos do *Bluetooth* são usuários de dispositivos móveis que necessitam estabelecer uma ligação, ou uma pequena rede, entre seus computadores, celulares e outros periféricos.

O alcance nominal do *Bluetooth* é estabelecido em 10m. Para suportar outras aplicações, como por exemplo, o ambiente de uma casa, o *chip Bluetooth* pode ser acrescido de um amplificador operacional externo para aumentar o alcance até 100m. Com um *hardware* auxiliar podem ser adicionados quatro ou mais canais de voz. Estas adições

ao *chip* base são totalmente compatíveis com a especificação nominal e podem ser adicionadas dependendo da aplicação.

Bluetooth utiliza uma rede WPAN chamada de *Ad hoc* com uma estrutura *piconet*. Duas ou mais *piconets* interconectadas formam uma s*catternet* como na Figura 7:

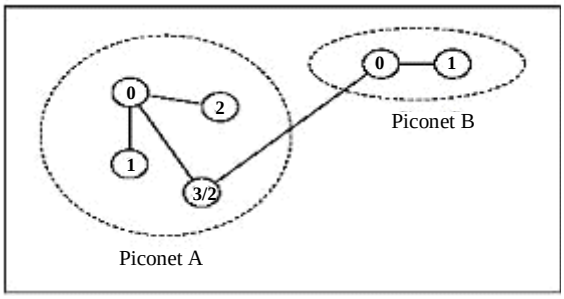

Figura 7: Exemplo *Scatternet* (HARRTSEN *et al*.,1998).

Numa rede *Bluetooth* todas as unidades têm hardware idêntico e as interfaces de software são distinguidas por um endereço de 48 bits. No início de uma conexão, a unidade que está inicializando é temporariamente designada como mestre. Esta designação é válida somente durante esta conexão.

O mestre que inicia a conexão e controla o tráfego sobre a conexão. Aos escravos são designados endereços temporários de 3 bits para reduzir o número de bits de endereçamento requeridos para comunicação. Uma rede *Bluetooth* suporta conexões ponto-a-ponto, envolvendo duas unidades *Bluetooth* apenas, e multiponto, envolvendo mais de duas unidades.

Bluetooth opera na freqüência de 2.4 GHz, transmitindo dados numa escala de 1Mbps, em média, com baixo consumo de energia para o uso em dispositivos móveis que necessitam de bateria para sua operação.

2.5 Protocolos de Serviços para Redes sem Fio

2.5.1 WAP

O WAP (*Wireless Application Protocol*) é um protocolo de comunicação que permite acessar conteúdos e serviços disponíveis na Internet através do telefone celular ou outros dispositivos móveis. Trata-se de um sistema que, através de um *microbrowser* une o poder

da rede às capacidades específicas dos dispositivos de comunicação celular (limite de caracteres na tela, capacidade de processamento e mobilidade) (HEIJDEN e TAYLOR, 2000). Um *microbrowser* ou *minibrowser* é um navegador web projetado para o uso em dispositivos tal como um PDA ou um celular que é otimizado para mostrar de forma eficaz o conteúdo da Internet para as pequenas telas dos dispositivos móveis.

O protocolo WAP foi publicado pela primeira vez em abril de 1998 pelo WAP Fórum – uma associação de fabricantes de dispositivos móveis, provedores de serviço e companhias de *software*, sendo fundado em julho de 1997 pela Ericsson, Motorola, Nokia e Phone.com. (HEIJDEN e TAYLOR, 2000).

O WAP está habilitado para as seguintes redes sem fio: CDPD, CDMA, GSM, TDMA, FLEX, ReFLEX, iDEN, TETRA, DECT, DataTAC, Mobitex, SMS, USSD, IS-136, etc, e utiliza um conjunto de protocolos de transmissão para transferir o conteúdo da Internet para os dispositivos dos usuários. A Figura 8 mostra estes protocolos de suporte.

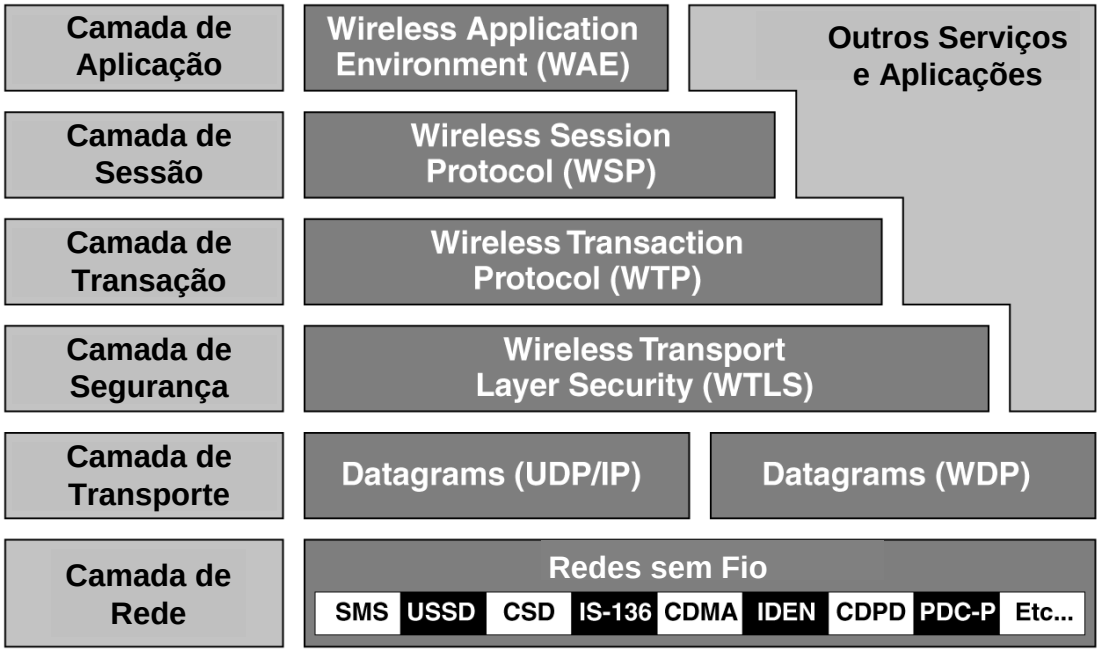

Figura 8: Pilha de Protocolos WAP. Adaptada de HEIJDEN e TAYLOR, 2000.

A base da pilha de protocolos é a camada de transporte chamada WDP (*Wireless Datagram Protocol*). O WDP oferece uma interface consistente com as camadas superiores da pilha.

A próxima camada é a de segurança WTLS (*Wireless Transport Layer Security*). Ela fornece um meio de transporte seguro entre o dispositivo WAP e o *gateway* WAP. WTLS

torna possível certificar-se que o conteúdo enviado não foi manipulado por terceiros. Também garante privacidade e garante que o autor da mensagem será identificado.

O WTP (*Wireless Transaction Protocol*) é a camada de transação responsável por controlar o envio e recepção de mensagens.

O WSP (*Wireless Session Protocol Layer*) é a camada de sessão que atua como uma interface entre a camada de aplicação a as outras camadas do modelo.

A camada mais alta é a de aplicação WAE (*Wireless Application Environment Layer*). Ela fornece um ambiente que permite o uso de uma quantidade muito grande de aplicações.

O conteúdo a ser acessado pelos dispositivos móveis deve ser desenvolvido em uma linguagem de marcação definida dentro do WAP chamada de WML (*Wireless Markup Language)* semelhante ao HTML (*Hypertext Markup Language),* porém é baseada em XML *(Extensible Markup Language).*

A linguagem WML é utilizada nas versões WAP 1.0, 1.1, 1.2 e 1.2.1, já na versão WAP 2.0, lançada em agosto de 2002, a linguagem XHTML (*Extensible HyperText Markup Language*), é utilizada. A linguagem XHTML é a sucessora do HTML (W3C, 2002).

Quando uma solicitação recebida é decodificada, o *gateway* executa a tarefa de converter a solicitação decodificada do protocolo WSP (no padrão WAP) para o protocolo da Internet HTTP (*Hypertext Transfer Protocol).*

O *gateway* WAP recebe o conteúdo WML do servidor Web e o converte para o padrão *bytecode* do WAP, codificando a informação para um formato binário de forma que possa ser usada menos largura de banda. Essa redução pode variar de 40 a 70%.

A Figura 9 mostra a arquitetura lógica do WAP.

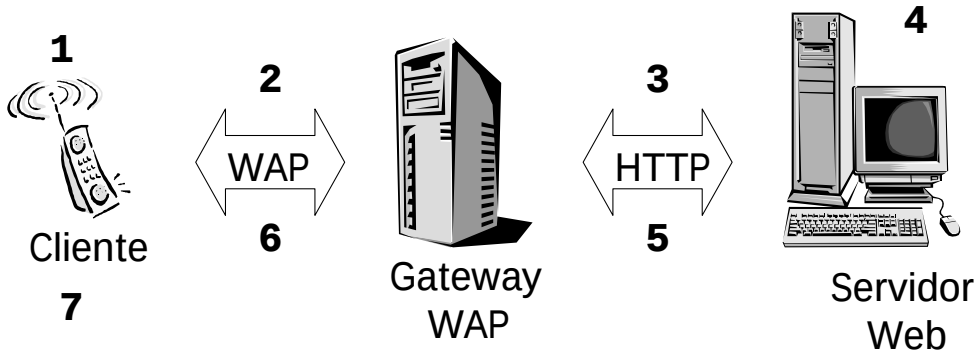

Figura 9: Arquitetura Lógica do WAP

2.5.2 I-mode

I-mode é uma plataforma proprietária para comunicação móvel que surgiu da necessidade vista pelo mercado japonês, representado pela companhia japonesa NTT DoCoMo, de implementar novos serviços no ramo da telefonia móvel. Essa nova forma de serviço móvel atraiu 45 milhões de assinantes desde seu lançamento em fevereiro de 1999 (Ntt DoCoMo, 2005a).

Com I-mode, celulares têm fácil acesso a várias páginas na Internet, além de outros serviços como: *e-mail*, compras *on-line*, acesso bancário, entretenimento, etc. Usuários podem acessar páginas de qualquer lugar do Japão e, independente do tempo de acesso, o sistema considera apenas o volume de dados transmitidos. A estrutura da rede NTT DoCoMo não somente provê acesso para o conteúdo *I-mode* compatível através da Internet, como também disponibiliza mecanismos de segurança e oferece voz e dados no mesmo pacote. A arquitetura de aplicação do *I-mode* é mostrada na Figura 10.

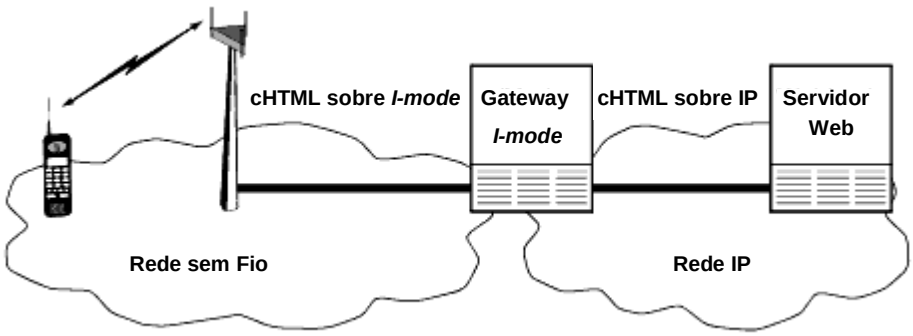

Figura 10: Arquitetura de Aplicação do *I-mode*.Traduzida de I-MODE, 2005.

O *gateway I-Mode* age como uma ponte entre a rede sem fio contendo os dispositivos móveis e a rede cabeada contendo o servidor de aplicações. Provedores de conteúdo criam páginas para *I-mode* usando a linguagem iHTML, também chamada de cHTML (*Compiled HTML*), que é uma versão reduzida do HTML. A companhia NTT DoCoMo define as *tags* e a estrutura de um documento cHTML.

2.5.3 SMS

O SMS (*Short Message Service*) surgiu em 1991 na Europa e hoje é um serviço globalmente aceito que habilita a transmissão de mensagens alfanuméricas entre os dispositivos móveis e do dispositivo móvel com sistemas externos como correio eletrônico,

paging, e sistemas de correio de voz. Este serviço está disponível sobre redes sem fio digitais que são baseadas nos padrões GSM, CDMA e TDMA (SMS, 2005).

Uma evolução do SMS é o MMS *(Multimedia Message Service)*, que permite a transmissão de mensagens multimídia contendo sons e imagens. Porém este serviço só está disponível para redes a partir da terceira geração.

2.5.4 Vodafone Live

O antigo protocolo *J-SkyWeb*, evoluiu para o serviço *Vodafone Live*, desenvolvido pela companhia japonesa Vodafone. Este serviço disponibiliza conteúdo em formato XHTML e XHTML MP (*Mobile Profile*).

O serviço *Vodafone Live* é semelhante, e ao mesmo tempo um concorrente, do *I-mode*, pois disponibiliza, além do acesso a Internet, praticamente os mesmos serviços da companhia Ntt DoCoMo, como *e-mail*, compras *on-line*, acesso bancário, entretenimento, etc. (VODAFONE, 2005).

2.5.5 EZweb

O serviço *EZweb* é mantido pela companhia japonesa AU, que é a parte de telefonia celular da operadora de longa distância KDDI e é similar aos serviços oferecidos pelas companhias NTT DoCoMo e Vodafone (KDDI, 2005).

As linguagens de apresentação utilizadas pelo EZweb são HDML (*Handheld Device Markup Language*) e XHTML do WAP 2.0.

2.6 Sumário

Este capítulo mostrou várias das tecnologias de redes sem fio, partindo de arquiteturas de rede e protocolos genéricos para as várias plataformas de rede, protocolos de serviços e ferramentas de desenvolvimento ora existentes no mercado.

Avaliando-se as tecnologias mostradas frente aos objetivos propostos no trabalho e considerando os requisitos de aplicações na área de *m-business*, pode-se concluir que:

- Das plataformas de comunicação sem fio para WWAN's, as que possibilitam a utilização de protocolos de serviços para a disponibilização de aplicações web são as de segunda e terceira gerações, onde a transmissão é digital. Na primeira geração a transmissão é analógica e centrada apenas em voz. Os dispositivos móveis que

utilizam esse tipo de plataforma geralmente são telefones celulares e dispositivos móveis como PDA's, *smartphones*, etc.

- Das plataformas de comunicação sem fio para WLAN's, o padrão 802.11 e suas extensões podem ser utilizadas visto que é análogo ao padrão *Ethernet;* apenas a conexão dentro da LAN é sem fio. A mobilidade se restringe à estação móvel estar "dentro" da WLAN. Geralmente, os dispositivos móveis utilizados nesse tipo de plataforma são *Palms, PDAs, Handhelds*, etc. Outros equipamentos, como *notebooks, laptops* e os computadores convencionais, também utilizam este tipo de plataforma para o acesso a aplicações web.
- Das plataformas de comunicação sem fio para WPAN's, o *Bluetooth* é a que se destaca e, por exemplo, pode ser utilizada por dispositivos móveis habilitados com a tecnologia para acessar aplicações web através de um adaptador *Bluetooth* USB conectado a um computador convencional que também possua uma forma de acesso à aplicação, seja por rede cabeada ou sem fio.
- Quanto aos protocolos de serviços, que geralmente são mantidos por operadoras de telefonia com redes de segunda e terceira gerações, pode-se afirmar que qualquer dispositivo móvel que possua qualquer um desses protocolos pode ter acesso à aplicações web, com exceção do SMS. Como o SMS serve apenas para a troca de mensagens curtas, ele pode ser utilizado como uma forma alternativa de envio de informação. Cada serviço tem suas particularidades, mas fazendo um estudo das características de cada um, o que pôde ser constatado é que cada protocolo ou serviço tem uma linguagem de apresentação diferente para mostrar o conteúdo. Essa questão é relevante no desenvolvimento da arquitetura proposta nesta dissertação pelo fato de que o objetivo é que ela seja genérica, ou seja, suporte qualquer tipo de protocolo e qualquer dispositivo móvel com suporte ao protocolo.

3 Tecnologias de Desenvolvimento Web

Neste capítulo serão apresentadas as principais tecnologias de desenvolvimento de aplicações web envolvidas nos trabalhos relacionados a seguir (Capítulo 4) para melhor entendimento das características de cada tecnologia visando também à aplicação na implementação da arquitetura de software proposta baseada em padrões de acordo com o objetivo proposto nesta dissertação.

3.1 Linguagens de Apresentação e Padrões Web

Como o foco é a disponibilização de uma aplicação web para a computação móvel, as linguagens de apresentação escolhidas foram WML e XHTML, que são as mais utilizadas no desenvolvimento de aplicações web para computação móvel no Brasil, já que predomina o protocolo WAP com o qual é possível acessar conteúdo disponibilizado nestas linguagens. O objetivo não é ensinar como desenvolver páginas em XHTML ou WML, mas sim dar uma visão geral da estrutura de cada linguagem.

Algumas linguagens já são definidas como padrões pelo W3C (*World Wide Web Consortium*), como no caso da linguagem XHTML que serve para a apresentação de páginas web, a linguagem XML que serve para o intercâmbio de informações e o padrão XSLT que serve para modificar o formato de apresentação de documentos XML. Cada linguagem será explicada separadamente a seguir.

3.1.1 XML

A XML é uma linguagem de marcadores criada pelo W3C com a intenção de se adicionar contexto e estrutura à informação contida na Web, que até então era feita usando HTML. Ela incorpora três características principais:

- **Extensibilidade**: XML possui a flexibilidade de permitir a definição de novos marcadores que tenham algum sentido para uma dada aplicação. Por esta característica, XML também pode ser vista como uma metalinguagem, pois permite definir outras linguagens de marcadores. Entretanto, o fato de não possuir um conjunto pré-definido de marcadores implica também em não haver semântica pré-definida. Toda semântica deve ser tratada pela aplicação que processa documentos XML.

- **Estrutura**: ao contrário da HTML, XML possui uma estrutura sintática mais rígida.
- **Validação**: a estrutura de um documento XML é definida formalmente por um *esquema*, que pode ser um DTD (*Document Type Definition*), ou um XML *Schema*. Assim, um documento XML pode ter sua estrutura validada.

Outra característica importante é o fato de que em XML é possível descrever os dados e os relacionamentos entre eles. Esta é uma das principais metas da XML: separar o conteúdo (dados), a estrutura (metadados) e a apresentação (a forma como os dados são apresentados).

XML é um padrão aberto, isto significa que é uma especificação "neutra" de fabricante, ou seja, documentos XML podem ser criados, modificados e processados por ferramentas de diferentes fabricantes que sigam a sua especificação.

Na Figura 11 é mostrado um exemplo simples de um documento XML. Ele inicia com a declaração XML: **<?xml version="1.0" encoding="ISO-8859-1"?>.**

Os elementos (*tags*) são os marcadores mais comuns da XML. O escopo de um elemento é definido por um *tag inicial* e um *tag final*. Na Figura x, o elemento `para` está definido entre os *tags* `<para>` e `</para>`. Note que o *tag inicial* é envolvido pelos caracteres "maior que" (<) e "menor que" (>), enquanto que o *tag final* possui, além destes caracteres, uma "barra invertida" (/) antes o nome do elemento.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<nota>
<de>Sara</de>
<para>Paulo</para>
<corpo> Não esqueça da água</corpo>
</nota>
```

Figura 11: Exemplo de um Documento XML

Existem outros elementos e particularidades relacionadas à linguagem XML que não serão abordados neste caso, pelo fato de que o objetivo é apenas dar uma visão geral ao leitor sobre o aspecto de cada linguagem citada.

3.1.2 WML

Como já foi dito anteriormente (seção 2.5.1), a linguagem WML é utilizada nas versões do protocolo WAP 1.0, 1.1, 1.2 e 1.2.1 e é semelhante ao HTML porém é baseada em XML.

A linguagem WML é composta por *tags* e seus atributos. A construção de páginas WML difere das construídas em HTML. Cada documento WML é chamado de *deck*, o qual pode conter um ou mais *cards*. Pode-se navegar entre os *cards* de um mesmo *deck* utilizando as teclas do dispositivo móvel. Um documento WML necessita, então, ser constituído por pelo menos um *card*. Dessa forma, os *links* de acesso podem ser direcionados para um *card* do *deck* ou para um outro documento WML.

Na figura 12 é mostrado um exemplo de um *deck* (página) WML constituído por dois *cards*. Como um *deck* WML é um documento XML ele inicia com a declaração XML: **<?xml version="1.0" encoding="ISO-8859-1"?>.** A DTD (*Document Type Declaration*) em negrito no documento WML se refere à linguagem WML. A seguir o documento WML possui o elemento **<wml>** que contém todos os outros subelementos e entidades usadas no documento e termina com **</wml>.** Assim como HTML as *tags* são envolvidas por <> e </>.

```
<?xml version="1.0" encoding= “ISO-8859-1?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<card id="WML" title="WML Tutorial">
<p>
Este é um tutorial sobre WML.
</p>
</card>

<card id="XML" title="XML Tutorial">
<p>
Este é um tutorial sobre XML.
</p>
</card>
</wml>
```

Figura 12: Exemplo de um *Deck* WML

A linguagem WML tem suporte à:

- **Layout e apresentação de texto:** embora a saída do código WML produzido varia entre os dispositivos e navegadores WML específicos (como acontece com as diferenças de saída existentes entre o *Netscape Navigator* e o *Internet Explorer*), a WML oferece suporte a quebras de linha, formatação de texto e alinhamento;

- **Imagens:** embora os dispositivos compatíveis com WAP não precisem oferecer suporte a imagens, a WML suporta o formato de imagem WBMP (*Wireless Bitmap*) e o alinhamento de imagens na tela.
- **Entrada do usuário:** a WML oferece suporte a listas de opções, listas de opções com vários níveis, entrada de texto e controles de tarefa;
- **Navegação:** o WAP oferece suporte aos *links* ancorados e ao esquema de nomenclatura de URLs da Internet padrão, permitindo a navegação entre *cards* em um *deck*, entre *decks* ou entre outros recursos na rede.

3.1.3 XHTML

A linguagem XHTML é uma recomendação do W3C, quase idêntica ao HTML 4.01. Ela é uma reformulação da linguagem de marcação HTML baseada em XML que combina as *tags* de marcação HTML com regras da XML. Esse processo de padronização visa à exibição de páginas Web em diversos tipos de dispositivos garantindo a acessibilidade.

Em XHTML os elementos (*tags*) devem ser aninhados corretamente, os documentos devem ser bem-formados, os elementos devem estar em letra minúscula e devem ser fechados. *Tags* vazias não são permitidas em XHTML. As tags <hr> e
 devem ser substituídas por <hr /> e
. Na Figura 13 é mostrado um exemplo simples da linguagem XHTML. Um documento XHTML consiste em três partes principais, o "**DOCTYPE**", o "**head**" e o "**body**". Ele pode ser definido por três tipos de documento (*Document Type Definitions*) que são: STRICT, TRANSITIONAL e FRAMESET. A mais comum é a TRANSITIONAL. Maiores detalhes no sítio do W3C (W3C; 2002)

```
<!DOCTYPE html
PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<title>Documento</title>
</head>
<body>
<p>Este é um parágrafo</p>
</body>
</html>
```

Figura 13: Exemplo de um Documento XHTML

3.1.4 XSLT

Uma das tecnologias Web utilizadas para gerar folhas de estilo padrão para a linguagem XML é a XSLT (*Extensible Stylesheet Language Transformation),* que é a parte mais importante dos padrões XSL (W3C; 2001).

A XSL *(Extensible Stylesheet Language*) consiste de três partes:

- XSLT – uma linguagem para a transformação de documentos XML;
- X-*Path* – uma linguagem para a navegação em documentos XML;
- XSL-FO – uma linguagem para a formatação de documentos XML.

Uma maneira de descrever o processo de transformação é dizer que XSLT transforma um arquivo fonte XML em um determinado arquivo de saída. A transformação é realizada pela associação do arquivo fonte com *templates* que são criados para determinar como será o arquivo de saída. Um XSL *Stylesheet* é um conjunto de *templates*, ou seja, um documento que contém as regras de apresentação para o formato desejado. Uma ilustração do processo de transformação pode ser visto na Figura 14.

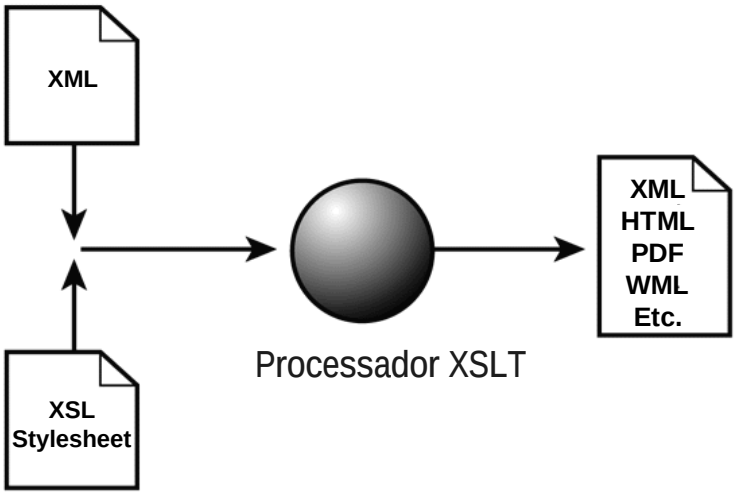

Figura 14: Processador XSLT

A estrutura do arquivo de saída pode ser completamente diferente do arquivo fonte. Na construção dos *templates*, elementos do arquivo fonte podem ser retirados ou reordenados e novas estruturas podem ser adicionadas (W3C, 2001).

Também para exemplificar o processo de transformação, serão mostrados um documento XML (Figura 15), um XSL *Stylesheet* criado para a linguagem WML (Figura 16) e o resultado para uma transformação para a linguagem de apresentação WML (Figura 18).

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!--Ordem de Compra SC1 -->
<OrdemDeCompra>
  <NumeroOrdem>CO-000001</NumeroOrdem>
  <DataOrdem>2005-05-20</DataOrdem>
</OrdemDeCompra>
```

Figura 15: Documento XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output doctype-public="-//WAPFORUM//DTD WML 1.1//EN" doctype-
system="http://www.wapforum.org/DTD/wml_1.1.xml" indent="yes"/>
<xsl:template match="/">
    <wml>
    <xsl:apply-templates/>
    </wml>
</xsl:template>
<xsl:template match="OrdemDeCompra">
<card title="Ordem de Compra">
<p>
<b><small>Número da Ordem: </small></b>
<br/>
<b> <xsl:value-of select="NumeroOrdem"/> </b>
<br/>
</p>
</card>
</xsl:template>
</xsl:stylesheet>
```

Figura 16: XSL *Stylesheet* para a linguagem de apresentação WML

Na Figura 16 o elemento raiz que declara o documento como um XSL *Stylesheet* é: **<xsl:stylesheet>** ou **<xsl:transform>.** Para ter acesso aos elementos e atributos XSLT, o *namespace* XSLT (Figura 17) deve ser declarado no topo do documento.

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
```

Figura 17: *Namespace* XSLT

Como um XSL *stylesheet* também é um documento XML ele inicia com a declaração XML: **<?xml version="1.0" encoding="ISO-8859-1"?>.**

A tag **<xsl:template>** define o início do *template*. O atributo **match="/"** associa o *template* com a *tag* raiz do documento fonte XML.

O resto do documento contém o *template*, exceto pelas duas últimas linhas que definem o fim do *template* e do XSL *Stylesheet*.

Com o Processador da Camada de Apresentação sendo baseado em XSLT pode-se transformar um documento XML em WML, cHTML, XHTML, etc e inclusive escolher quais elementos são relevantes de serem visualizados de acordo com tipo de dispositivo escolhido no momento da criação dos *templates* como pode ser observado na Figura 18, na qual o elemento **<DataOrdem>** do documento XML da Figura 15 não foi escolhido. Sendo assim ele não irá aparecer no arquivo de saída, como pode ser observado na Figura 18.

A DTD (*Document Type Declaration*) em negrito no *template* (Figura 16) e no documento WML gerado (Figura 18) se refere à linguagem WML. Cada linguagem de apresentação possui uma DTD e esta deve ser especificada no *template*.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">
<wml>
<card title="Ordem de Compra">
<p>
<b><small>Número da Ordem: </small></b>
<br/>
<b>CO-000001</b>
<br/>
</p>
</card>
</wml>
```

Figura 18: Documento WML gerado.

Um ponto importante a ser tratado para a utilização do padrão XSLT é de que os dados devem estar no formato XML para que possam ser associados com os *templates* criados.

3.2 Plataformas de Desenvolvimento de Aplicações para Dispositivos Móveis

3.2.1 BREW

A plataforma proprietária BREW (*Binary Runtime Enviroment for Wireless*), desenvolvida pela QUALCOMM surgiu no fim de 2001 com o objetivo de habilitar em dispositivos móveis os mesmos tipos de aplicações que hoje existem somente para

desktops. Aplicações em potencial incluem *e-mail*, serviços de navegação em tempo real, *chat*, jogos em grupo, serviços de informação, etc (BREW, 2004a).

O BREW proporciona, para o desenvolvimento de aplicações, as características providas pelos *chipsets* CDMA da QUALCOMM. A vantagem da plataforma é que ela fica situada entre o *chip* de software do sistema e a aplicação, permitindo-se criar aplicações compatíveis com os *chipsets* CDMA sem requerer conhecimento do código fonte do *chip* de sistema. Também permite escrever aplicações independentes de hardware, de rede e que podem ser instaladas em uma variedade de telefones celulares, sem modificação da aplicação para cada novo dispositivo ou modelo (BREW, 2004b).

BREW usa C e C++ como linguagens para desenvolvimento, e também suporta a integração de aplicações Java se uma Máquina Virtual Java está disponível no dispositivo. Especificações técnicas e um SDK (*Software Development Kit*) baseado em Windows do BREW estão livremente disponíveis no *Web site*, mas somente sob registro com a QUALCOMM.

3.2.2 J2ME

J2ME (*Java2 Micro Edition*) é uma arquitetura padrão e aberta, para o desenvolvimento de aplicações para dispositivos móveis, criada pela Sun Microsystems. Tal tecnologia consiste na simplificação da máquina virtual Java, de forma a mesma melhor se adequar à limitação de recursos (memória, capacidade de processamento, bateria) disponíveis nos dispositivos móveis. J2ME proporciona ao desenvolvedor o uso de uma linguagem de programação, fazendo com que possam ser criados aplicativos que sejam instalados, por exemplo, via *download* através de um cabo serial conectado a um PC (*Personal Computer*), ou via a própria rede de dados dos celulares.

Diferente de *desktops* e servidores, alvos do J2SE (*Java 2 Standard Edition)* e J2EE (*Java 2 Enterprise Edition)*, o ambiente *Wireless* inclui uma gama de dispositivos com capacidades imensamente diferentes que não é possível criar um único *software* para todos deles. Então, em vez de ser uma única entidade, J2ME é uma coleção de especificações que definem um conjunto de plataformas, cada qual satisfatória para um subconjunto do total de dispositivos. O subconjunto do ambiente de programação Java para um dispositivo em particular está definido por um ou mais perfis que estendem as capacidades básicas de uma configuração. A configuração e o perfil (ou perfis) que são apropriados para um dispositivo depende de seu hardware e da sua finalidade (TOPLEY, 2002).

É importante afirmar que, tanto o J2ME como o BREW, são plataformas para o desenvolvimento de aplicações para a serem instaladas no lado cliente, ou seja, para serem executadas nos diferentes tipos de dispositivos móveis existentes.

3.2.3 J2EE

O J2EE (*Java 2 Enterprise Edition*) ou Java EE é uma plataforma de programação de computadores que faz parte da plataforma Java. Ela é voltada para aplicações multicamadas desenvolvidas na linguagem Java baseadas em componentes que são executados em um servidor de aplicações (JAVA EE, 2006). A plataforma Java EE é considerada um padrão de desenvolvimento incluindo bibliotecas desenvolvidas para o acesso a base de dados, RPC (*Remote Procedure Call*), Corba, etc. Devido a essas características a plataforma é utilizada principalmente para o desenvolvimento de aplicações corporativas.

A plataforma J2EE contém uma série de especificações, cada uma com funcionalidades distintas. Entre elas, tem-se:

- JDBC (*Java Database Connectivity*), utilizado no acesso a bancos de dados;
- *Servlets*, utilizados para o desenvolvimento de aplicações Web com conteúdo dinâmico. Eles contêm uma API que abstrai e disponibiliza os recursos do servidor Web de maneira simplificada para o programador.
- JSP (*Java Server Pages*), uma especialização dos *Servlets* que permite que conteúdo dinâmico seja facilmente desenvolvido.
- JTA (*Java Transaction API*), é uma API que padroniza o tratamento de transações dentro de uma aplicação Java.
- EJBs (*Enterprise JavaBeans*), utilizados no desenvolvimento de componentes de *software*. Eles permitem que o programador se concentre nas necessidades de negócio do cliente, enquanto questões de infra-estrutura, segurança, disponibilidade e escalabilidade são de responsabilidade do servidor de aplicações.

3.2.4 Servlets

O cliente solicita que alguma ação seja realizada e o servidor realiza a ação e responde para o cliente. Esse modelo de comunicação solicitação-resposta é o fundamento

para a visualização do mais alto nível de redes em Java – *Servlets*. Um *servlet* estende a funcionalidade de um servidor. O pacote **javax.servlet** e o pacote **javax.servlet.http** fornecem as classes e as interfaces para definir os *servlets*.

A tecnologia de *servlet* hoje é projetada principalmente para a utilização com o protocolo HTTP da *World Wide Web*, mas estão sendo desenvolvidos *servlets* para outras tecnologias. Os *servlets* são eficientes para desenvolver soluções baseadas na Web que ajudam a fornecer acesso seguro a uma página da Web, interagir com banco de dados, gerar dinamicamente documentos personalizados a serem exibidos por navegadores e manter informações para sessão exclusivas de cada cliente (DEITEL e DEITEL, 2001).

Muitos desenvolvedores acham que os *servlets* são a solução certa para aplicações de uso intenso de banco de dados que se comunicam com os chamados *thin clients* – aplicativos que exigem suporte mínimo no lado do cliente. O servidor é responsável pelo acesso ao banco de dados. Os clientes conectam-se ao servidor utilizando protocolos padrão disponíveis em todas plataformas clientes. Portanto o código lógico pode ser escrito uma vez e residir no servidor para ser acessado pelos clientes (DEITEL e DEITEL, 2001).

Os pacotes de *servlet* definem duas classes **abstract,** que implementam a interface **Servlet –** a classe **GenericServlet** (do pacote **javax.servlet**), e a classe **HttpServlet** (do pacote **javax.servlet.http**). Essas classes fornecem implementações padrão de todos os métodos de **Servlet.** A maioria dos servlets estende **GenericServlet** e **HttpServlet** e sobrescreve alguns ou todos os seus métodos com comportamentos personalizados adequados.

3.2.4.1 A classe HttpServlet

Servlets baseados na Web, em geral, estendem a classe **HttpServlet**. A classe **HttpServlet** sobrescreve o método **service** para distinguir entre as solicitações típicas recebidas de um navegador Web cliente. Os dois tipos mais comuns de solicitação HTTP (também conhecidos como métodos de solicitação) são **GET** e **POST**.

Uma solicitação **GET** obtém (ou busca) informações do servidor. As utilizações comuns de solicitações **GET** são buscar um documento HTML ou uma imagem. Uma solicitação **POST** posta (ou envia) dados para o servidor. As utilizações comuns de solicitações **POST** consistem em enviar ao servidor informações de um formulário HTML em que o cliente insere dados, enviar informações ao servidor para que este possa pesquisar na Internet ou consultar um banco de dados para o cliente, etc.

A classe **HttpServlet** define os métodos **doGet** e **doPost** para responder as solicitações **GET** e **POST** de um cliente, respectivamente. Esses métodos são chamados pelo método **service** da classe **HttpServlet**, que é chamado quando uma solicitação chega no servidor. O método **service** primeiro determina o tipo de solicitação, então chama o método apropriado.

Os métodos **doGet** e **doPost** recebem como argumentos um objeto **HttpServletRequest** que implementa a interface **HttpServletRequest** e um objeto **HttpServletResponse** que implementa a interface **HttpServletResponse**, que permitem interação entre o cliente e o servidor. Os métodos de **HttpServletRequest** tornam fácil acessar os dados fornecidos como parte da solicitação. Os métodos **HttpServletResponse** tornam fácil retornar os resultados do *servlet* no formato desejado para o cliente da Web.

3.2.5 JSP

JSP ou *Java Server Pages* é uma extensão da tecnologia de *Servlets* em Java utilizada para criar páginas Web que mostram conteúdo gerado dinamicamente. Em JSP páginas são criadas de maneira parecida com as que são criadas em ASP (*Active Server Pages*) ou PHP (*Hypertext Processor*), outras duas tecnologias de servidor.

Uma página JSP é uma página que inclui tecnologia JSP específica e *tags* especiais (*scriplets*) escritas na linguagem de programação Java combinadas com *tags* estáticas (HTML ou XML, por exemplo). A especificação JSP define a interação entre o servidor e a página JSP e descreve o formato e a sintaxe da página. Uma página JSP tem a extensão .jsp ou .jspx; isto sinaliza ao servidor web que a máquina de JSP processará elementos nesta página. Na Figura 19 é mostrado um exemplo simples de uma página JSP para a linguagem HTML onde as *tags* JSP estão em negrito.

```
<html>
  <head>
    <title>Exemplo JSP</title>
  </head>
  <body>
    <%
      String x = "Ola Mundo!"
    %>
     <%=x%>
  </body>
</html>
```

Figura 19: Exemplo de Página JSP

3.3 Sumário

Este capítulo apresentou algumas das principais tecnologias e padrões envolvidos no desenvolvimento de aplicações web para dispositivos móveis atualmente. É importante ressaltar que existem várias tecnologias disponíveis para o desenvolvimento de aplicações e apenas algumas delas foram citadas. Estas foram escolhidas baseando-se nos trabalhos relacionados a seguir e no objetivo proposto de se utilizar tecnologias abertas e baseadas em padrões para o desenvolvimento.

4 Trabalhos Relacionados

Neste capítulo serão apresentados alguns trabalhos relacionados mostrando soluções proprietárias, trabalhos acadêmicos, e projetos de pesquisa na área de redes sem fio onde o principal objetivo é disponibilização de aplicações web para dispositivos móveis.

4.1 Soluções Proprietárias

4.1.1 Mobile Internet Platform 5.0

A empresa *Air2Web* em Atlanta, nos EUA, lançou a *Mobile Internet Platform 5.0*, que é uma plataforma de desenvolvimento para a criação de aplicações móveis em áreas como EAI (*Enterprise Application Integration),* ERP *(Enterprise Resource Planning),* e SCM (*Supply Chain Management*). Utilizando padrões abertos como J2EE e XML, é possível desenvolver aplicações independentes de protocolo, tecnologia de rede sem fio e dispositivo móvel utilizado (AIR2WEB, 2003).

Esta é uma solução que, apesar de proprietária, utiliza padrões abertos como J2EE e XML para o desenvolvimento das aplicações. Isso irá proporcionar ao cliente que adquirir a solução, o desenvolvimento de determinada aplicação de acordo com o tipo de dispositivo móvel escolhido.

A abordagem para o desenvolvimento de aplicações utilizada pela plataforma é baseada em padrões abertos e poderia ser aproveitada no desenvolvimento da arquitetura proposta já que a utilização desses padrões facilita a interoperabilidade entre dispositivos.

4.1.2 WebSphere Transcoding Publisher

Um dos produtos da IBM é o *WebSphere Transcoding Publisher,* que é um software que fica no servidor e transfere o conteúdo Web e de outras aplicações para várias linguagens de apresentação, tornando possível o acesso a informações para vários tipos de dispositivos móveis, como telefones celulares, *handhelds, palms,* etc (IBM, 2005).

Sendo assim, a solução da IBM possibilita ao cliente que adquiri-la a disponibilização de informações para as várias linguagens de apresentação existentes, portanto é independente do protocolo de comunicação e do tipo de dispositivo utilizado.

A solução proposta pela IBM se encaixa dentro do objetivo principal deste trabalho e poderia ser utilizada apenas em termos de concepção da arquitetura de software proposta já que é uma solução proprietária.

4.1.3 HAND BUSINESS

Beneficiando-se da parceria com operadoras de telefonia celular**,** a TWIC desenvolveu sua linha de produtos para computação móvel denominada HAND BUSINESS. A linha HAND BUSINESS permite conexão em tempo integral entre um PDA e servidores corporativos para qualquer banco de dados, em qualquer plataforma e em qualquer lugar, além de permitir o uso de ferramentas de escritório como *e-mails*, planilhas, processadores de texto, envio e recepção de mensagens (TWIC, 2005).

Neste caso, a solução completa é fornecida pela empresa, não permitindo a flexibilidade do cliente escolher qual o tipo de dispositivo móvel e qual operadora de telefonia vai utilizar. Portanto, não está dentro dos objetivos de propor uma arquitetura independente de dispositivo móvel para que a escolha do mesmo fique a cargo do cliente.

4.1.4 Enterprise Software

A *Enterprise Software* é uma empresa de consultoria, planejamento e desenvolvimento de soluções de mobilidade corporativa que desenvolve aplicações de negócios personalizadas totalmente integradas aos Sistemas de Gestão Empresarial e a outros sistemas legados de seus clientes. As linguagens de programação utilizadas são Super Waba e J2ME para os PDA’s e J2EE para as aplicações no servidor Web. Os sistemas utilizados são PalmOS, Pocket PC e WinCE (ENTERPRISE SOFTWARE, 2005).

Porém, pelo que pode ser percebido, há limitações quanto ao tipo e também quanto ao sistema operacional dos dispositivos móveis; sendo assim, um cliente que resolver adotar a solução fornecida pela empresa terá que adquirir dispositivos compatíveis com as aplicações fornecidas.

4.1.5 Teleca Enterprise Mobility

A Teleca é uma companhia internacional na área de tecnologia da informação que desenvolve e integra soluções para as mais variadas áreas, como saúde, transporte, comércio, manufatura, etc. Na área de redes sem fio a empresa possui um *framework* chamado *Teleca Enterprise Mobility*. Esse *framework* tem suporte para os mais variados

tipos de dispositivos móveis, tais como: Symbian, J2ME, Pocket PC, etc. O acesso à informação pode ser feito através do WAP, SMS, Web, etc (TELECA, 2005).

Portanto, o *framework* da Teleca permite o desenvolvimento de aplicações no lado cliente e no lado servidor e também tem suporte para vários tipos de dispositivos e meios de acesso à informação; porém não será aproveitado pelo fato de ser proprietário e não mostrar as tecnologias envolvidas no desenvolvimento do mesmo.

4.2 Projetos de Pesquisa

4.2.1 Projeto MACRO

O projeto MACRO (*Mobility Assistance for Customer Relations Based Organizations*) teve como objetivo examinar questões técnicas e paradigmas de negócios nos softwares para CRM (*Customer Relationship Management*), ERP (*Enterprise Resources Planning*) e SCM (*Supply Chain Management*) quando estes são conectados via Web e a tecnologias de redes sem fio.

O projeto criou uma plataforma de software utilizando a tecnologia de *Web Services* para atuar como um ambiente de integração entre os sistemas de CRM, SCM e ERP e fazer os dados contidos nesses sistemas disponíveis para dispositivos móveis de acordo com a estratégia móvel global de determinada companhia (MACRO, 2005).

De acordo com a tecnologia de *Web Services* é necessário ter uma aplicação no dispositivo móvel, ou seja, no lado cliente, que suporte *Web Services* e utilize os serviços disponibilizados pelo sistema.

Este projeto, apesar de disponibilizar dados para dispositivos móveis, não se encaixa dentro da arquitetura de software proposta pois leva em consideração o tipo de dispositivo do cliente, e este deve ter suporte a *Web Services*.

4.2.2 Projeto *Mobile Commerce*

O projeto *Mobile Commerce* da Universidade Européia Viandrina de Frankfurt, na Alemanha, vem estudando a utilização do protocolo WAP em sistemas ERP.

Para tornar possível o acesso à base de dados do sistema ERP para os dispositivos móveis os dados devem ser convertidos para WML (*Wireless Markup Language*). Da

mesma forma, os dados enviados pelos dispositivos móveis devem ser convertidos para a visualização no sistema ERP (MOBILE COMMERCE, 2005).

O Projeto *Mobile Commerce* se limita apenas à pesquisa da utilização do protocolo WAP sem se preocupar com os vários protocolos e possibilidades de acesso à informação através de dispositivos móveis, diferente da arquitetura proposta, a qual pretende ser independente do protocolo de comunicação utilizado pelo dispositivo móvel.

4.2.3 Projeto WILMA

O projeto WILMA (*Wireless Internet and Location Management Architecture*), do departamento de Informática e Telecomunicações da Universidade de Trento, na Itália, estuda soluções para a integração de diferentes protocolos e meios de acesso sem fio para suportar *Pervasive Computing. Pervasive computing* é a criação de ambientes com computação e comunicação sem fio integradas com os usuários (IEEE, 2006).

As pesquisas incluem migração de informação entre diferentes ambientes, busca de meios de acesso sem fio para contextos específicos, antecipação de requisições de usuário baseadas em histórico, etc (WILMA, 2001).

No projeto WILMA há um estudo mais aprofundado que leva em consideração protocolos e meios de acesso sem fio e envolve várias tecnologias de desenvolvimento dando ênfase para mecanismos de autenticação, segurança, QoS, etc, utilizados na troca de informações, diferente deste trabalho no qual o foco está na disponibilização de uma aplicação web para dispositivos móveis, sem se preocupar com aspectos de segurança e QoS.

4.2.4 Projeto WAPEDUC

O projeto WAPEDUC iniciado pela *Académie de Montpellier* - França, tem como objetivo disponibilizar testes e recursos aos alunos através dos seus celulares. Já há mais de 600 questionários de avaliação e recursos que foram selecionados e criados de acordo com as restrições curriculares. Para acessar através de um celular WAP digitar a URL (*Uniform Resource Locator)*: http://wapeduc.net, e de um celular I-mode: http://www.mobileduc.net. (WAPEDUC, 2005).

O projeto WAPEDUC é um projeto, mas também uma aplicação que disponibiliza informações escolares em WAP e *I-mode,* mostrando certa flexibilidade no fornecimento

das informações, porém não menciona de que maneira o mesmo conteúdo é disponibilizado em dois protocolos diferentes.

4.2.5 Open Mobile Alliance

Apesar de não ser um projeto, é importante mencionar a iniciativa OMA (*Open Mobile Alliance*). Ela é uma organização que tem a missão de facilitar ao usuário global a adoção de serviços de dados móveis assegurando a interoperabilidade entre dispositivos independente de localização, de provedores de conteúdo, de operadoras e de redes sem fio. Praticamente todas as grandes empresas que desenvolvem serviços, dispositivos e aplicações na área de telefonia e tecnologia da informação fazem parte dessa aliança (OMA, 2005).

Esta organização está desenvolvendo especificações em conjunto com as empresas para facilitar a interoperabilidade, por isso é importante prestar atenção nas especificações e novas tecnologias de desenvolvimento para acompanhar a evolução do mercado.

4.3 Trabalhos Acadêmicos

4.3.1 Sistema de Monitoramento Remoto

Em KIMURA e KANDA (2003) um sistema de monitoramento remoto foi desenvolvido utilizando um telefone portátil para operações colaborativas entre uma pequena empresa e o lado remoto, como um componente de um sistema de suporte à manufatura. Os terminais deste sistema são telefones celulares e pelo fato de os usuários nem sempre possuírem um telefone celular com a função de navegador, o método de comunicação básico do sistema é por *E-mail*, limitado pelo registro de endereço de *E-mail*, não permitindo a verificação dos dados *on-line*. Algumas funções do sistema são: Confirmação do *status* de produção por nome de produto; Notificação quando um atraso na produção ocorre e Notificação quando ocorre problema com algum equipamento.

Para a arquitetura proposta, um dos requisitos mínimos necessários é o de que o dispositivo móvel deve possuir um navegador web, já que uma aplicação web será disponibilizada e, em KIMURA e KANDA (2003) o método de comunicação é limitado, apenas por *E-mail*, não permitindo a verificação dos dados *on-line*, o que será possível dentro da arquitetura proposta.

4.3.2 Técnicas para o Desenvolvimento de Aplicações Globais

Em DABKOWSKI *et al*. (2003) são utilizadas duas técnicas para o desenvolvimento de aplicações globais utilizando JSP (*Java Server Pages*): JSP *Tag Libraries* e XSLT (*Extensible Stylesheet Language Transformations*). Aplicações globais são desenvolvidas independente dos países ou linguagem dos usuários e ajustadas para os requisitos dos vários países ou regiões. Nesse caso as aplicações são via Internet para clientes de *desktop* e de dispositivos móveis.

Dependendo do tipo de dispositivo e configurações de linguagem, os mesmos dados podem ser apresentados na linguagem do usuário em um computador com um navegador *Internet Explorer* ou em celular com um navegador WAP, por exemplo.

Este trabalho utiliza técnicas de transformação como XSLT, que é um padrão aberto, para tornar a aplicação genérica para os vários tipos de dispositivos e ainda utiliza recursos do Java para tornar a aplicação global.

Ele foi utilizado como base para o desenvolvimento da arquitetura de software proposta, pois possibilita a uma aplicação web, do lado servidor, ser disponibilizada para a Computação Móvel de forma a poder ser acessada por um cliente, independente do tipo de dispositivo utilizado e ainda utiliza padrões abertos para o desenvolvimento.

A diferença entre esse trabalho e a arquitetura de software proposta está na abordagem de desenvolvimento. No primeiro, utiliza-se páginas JSP para a lógica de apresentação, o que não é apropriado para a proposta em questão pois o JSP é mais adequado para apresentação de páginas web. A arquitetura de software proposta utiliza *Servlets*, que são apropriados para o controle da aplicação web, separando as camadas lógica e de apresentação, potencializando uma mais fácil reutilização.

4.3.3 Método para a Geração de *XSL Stylesheets*

Em KWOK *et al*. (2004) é proposto um método para a geração de XSL *Stylesheets* para diferentes dispositivos móveis. Um *template* é fornecido ao usuário para que ele entre com as informações do dispositivo como: tamanho de tela, número de linhas, número de caracteres por linha e funções suportadas. Com essas informações, um gerador de regras de apresentação, constrói o novo conjunto de regras para o dispositivo determinado.

O trabalho de KWOK *et al*. (2004) solicita informações do usuário para que ele possa, através de um conjunto de regras de apresentação fornecer a informação adequada; portanto está sujeito a erros por parte do usuário final.

Como em DABKOWSKI *et al*. (2003), KWOK *et al*. (2004) também utiliza o padrão XSLT para gerar conteúdo para os diferentes tipos de dispositivos móveis, e este padrão também será utilizado como tecnologia de desenvolvimento da arquitetura de software proposta.

4.3.4 DIGESTOR

O DIGESTOR é um sistema que recria documentos da Internet para mostrá-los de forma apropriada em dispositivos com telas pequenas como PDAs e celulares, disponibilizando o acesso à Internet independente de dispositivo. Ele é implementado como um *proxy* HTTP que recria dinamicamente páginas da Internet solicitadas utilizando um algoritmo baseado em regras e um conjunto de estruturas de transformação de páginas para melhor apresentação do documento para um dado tamanho de tela (BICKMORE e SCHILIT, 1997).

Este sistema também solicita como dados do usuário o tamanho da tela do dispositivo, a fonte *default* do navegador e a URL desejada. Além de estar sujeito a erros por parte do usuário, é baseado em regras de transformação e se surgir algo não previsto dentro destas regras, o resultado pode estar fora do esperado.

Na arquitetura proposta o usuário não precisará entrar com dados técnicos para que a disposição do conteúdo na tela seja adequada uma vez que isto já é previsto e feito no momento do desenvolvimento da aplicação web.

4.3.5 Interfaces Adaptáveis

Em NYLANDER e BYLUND (2002) é apresentada uma visão para a criação de serviços eletrônicos com interfaces de usuário adaptadas para as diferentes necessidades dos usuários e as diferentes capacidades dos dispositivos. Um conjunto de “atos de interação” é utilizado para descrever a interação do usuário com o serviço acessado e esta é complementada com a descrição do dispositivo e informações específicas do usuário e, com isso, uma máquina de interação gera a interface apropriada para o dispositivo que solicitou o serviço.

Este trabalho utiliza algoritmos para adaptar o conteúdo e solicita informações do usuário para gerar uma interface apropriada de acordo com o dispositivo que solicitou a informação, porém não considera apenas o formato de apresentação, mas também o tipo de usuário, como, por exemplo, um deficiente visual, que necessita de outro tipo de interface com o usuário. Portanto, este trabalho considera o "tipo" de usuário que utiliza o dispositivo, o que não é levado em consideração na arquitetura proposta.

4.3.6 DDL

Em GÖBEL *et al.* (2001) é apresentada uma linguagem de descrição independente de dispositivo (DDL) para diálogos e conteúdos baseados em Web. Juntamente com essa linguagem foi criado um *framework* de adaptação que processa as descrições DDL para gerar diálogos para dispositivos específicos.

Neste trabalho é adotada uma nova técnica que ao invés de utilizar padrões já existentes, cria uma nova linguagem para disponibilizar conteúdo Web independente de dispositivo, o que pode ser um caminho para acabar com a diversidade de linguagens existentes para cada protocolo de comunicação, porém, há longo caminho a percorrer até a padronização.

Já a arquitetura proposta disponibiliza conteúdo web para qualquer linguagem de apresentação de conteúdo Web existente, basta que sejam criados estilos de apresentação de acordo com a linguagem desejada.

4.3.7 Ferramentas de Busca

Em *Sonera Medialab*, um laboratório da companhia de telecomunicações TeliaSonera, na Finlândia, é apresentada uma arquitetura para o desenvolvimento de ferramentas de busca para dispositivos móveis. A arquitetura consiste de um navegador, um *parser*, um indexador, um mecanismo de busca e interfaces para as diferentes linguagens de apresentação como WML, HTML, cHTML, etc. De acordo com a arquitetura apresentada os resultados são mostrados para o usuário no formato adequado do dispositivo que realizou a busca. Alguns exemplos de ferramentas de busca são Google, *FAST, WithAir, Pinpoint, IndexCell*, etc (SONERA MEDIALAB, 2002).

Comparando este trabalho com a arquitetura proposta nesta dissertação, pode-se afirmar que a proposta trata apenas da parte das interfaces para as diferentes linguagens de

apresentação existentes; toda lógica que existe por trás de uma ferramenta de busca não é considerada.

4.4 Conclusões

Avaliando os diversos trabalhos existentes pode-se verificar que há diferença entre soluções proprietárias e trabalhos acadêmicos, pois um é direcionado para as tendências do mercado sem se preocupar com o grau de abrangência de dispositivos que podem ser alcançados, e o outro para a descoberta de novos métodos que possam abranger todas as classes de dispositivos móveis.

Examinando as soluções proprietárias pode-se perceber que há limitações quanto aos tipos, ao sistema operacional, ao protocolo de serviço e a marca dos dispositivos utilizados.

No caso das empresas que fornecem *framework* de desenvolvimento, geralmente o cliente adquire a solução completa (portanto tendendo a ser mais cara do que seria o necessário) e a implanta na empresa, e nem sempre todo potencial da solução é utilizado, pois uma solução deste tipo é desenvolvida pensando-se nos vários tipos de clientes que possivelmente irão adquiri-la e o pacote é fechado e vendido aos clientes sem distinção.

Quanto aos trabalhos e projetos acadêmicos, pode-se verificar que há pesquisa utilizando diferentes métodos e técnicas para tornar as aplicações web disponíveis para os diversos tipos de dispositivos móveis existentes.

Existem iniciativas das grandes empresas de telefonia e de tecnologia da informação, na tentativa de criar uma arquitetura móvel aberta, mas na prática, isto é, através dos produtos fornecidos por essas empresas, a interoperabilidade ainda está longe de ser alcançada.

Soluções proprietárias como da empresa *Air2Web* e IBM e trabalhos acadêmicos como o de DABKOWSKI *et al*. (2003) e KWOK *et al*. (2004) serviram de base para propor alguns componentes da arquitetura que será explicada no capítulo 5, pois utilizam uma abordagem de desenvolvimento e padrões abertos, como J2EE, XML e XSLT, que facilitam a disponibilização de aplicações web para os vários tipos de dispositivos móveis existentes.

5 Arquitetura Proposta

5.1 Introdução

Após uma avaliação das tecnologias de redes sem fio e dos trabalhos relacionados, este capítulo apresenta a arquitetura proposta com todos os componentes vistos para que se viabilize a disponibilização de uma aplicação web para dispositivos móveis considerando os objetivos desejados (expostos na seção 1.2).

A arquitetura genérica de software proposta é mostrada na Figura 20 e cada componente será explicado em detalhes. Alguns componentes da arquitetura foram inspirados na obra de BRANS (2002), onde é mostrado o que é necessário e o que precisa ser feito para que uma empresa possa adotar soluções móveis. No entanto, nesta obra nada é proposto em termos de uma arquitetura propriamente dita, mas a maneira como cada parte é descrita facilita a visão integrada dos componentes e a visualização destes.

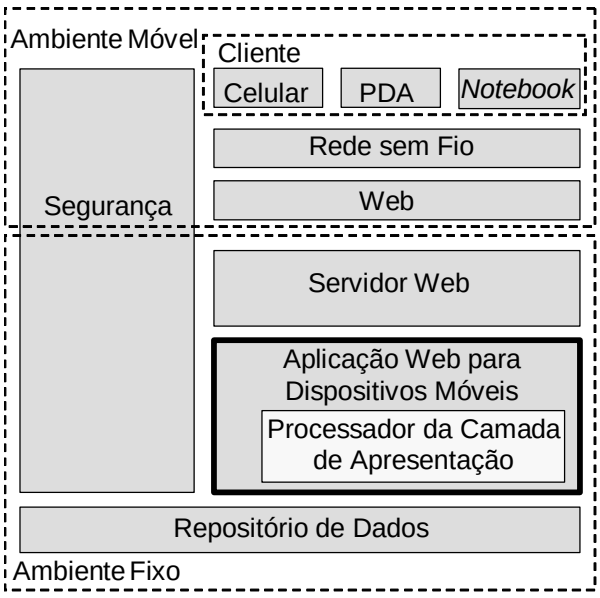

Figura 20: Arquitetura Proposta

Um trabalho que serviu de base para conceber a arquitetura foi o de DABKOWSKI *et al*. (2003), que mostra uma arquitetura para aplicações globais para diferentes tipos de dispositivos. Porém, neste trabalho a arquitetura está idealizada já se indicando exatamente

as tecnologias de informação utilizadas para o desenvolvimento da aplicação web a ser disponibilizada.

Já para a arquitetura proposta é apresentado primeiramente um modelo conceitual sem considerar as tecnologias envolvidas no desenvolvimento, e depois, na implementação estas são mostradas, visto que existem várias tecnologias para o desenvolvimento da aplicação web e a escolha depende do projeto em questão.

5.2 Componentes da Arquitetura

Nesta seção serão explicados todos os componentes da arquitetura proposta para a disponibilização de uma aplicação web para dispositivos móveis mostrando a função de cada um dentro da arquitetura.

A arquitetura foi dividida em Ambiente Fixo e Ambiente Móvel para facilitar a visualização dos componentes e também para mostrar que é no Ambiente Fixo, ou seja, no lado servidor que são feitas modificações para tornar a aplicação web disponível para os diversos tipos de dispositivos.

5.2.1 Repositório de Dados

O Repositório de Dados é responsável pela manutenção e integridade dos dados e informações da aplicação, que pode ser implementada de forma centralizada com apenas um SGBD (Sistema Gerenciador de Banco de Dados) ou de forma distribuída com mais de um SGBD em servidores diferentes.

É do repositório de dados que a Aplicação Web retira todas as informações a serem apresentadas. Dependendo do banco de dados utilizado e da tecnologia de desenvolvimento da Aplicação Web, uma conversão de formato é feita nos dados do repositório para que estes possam ser acessados de acordo com as necessidades da aplicação web.

5.2.2 Aplicação Web para Dispositivos Móveis

A Aplicação Web para Dispositivos Móveis não deixa de ser uma aplicação web comum, ou seja, um programa disponibilizado por um Servidor Web que é utilizado por meio de um navegador Web.

No entanto, como já foi dito anteriormente, para tornar as aplicações web disponíveis também para os dispositivos móveis, segundo o modelo *m-business*, é necessária uma adaptação no conteúdo para que ele possa ser acessado por equipamentos com telas e teclados menores e com memória e poder de processamento limitados.

Outro ponto importante a ser considerado no desenvolvimento da aplicação web é o tipo de informação a ser disponibilizado para a Computação Móvel onde o cliente quer acesso rápido, simples e com conteúdo especializado, diferente do modo fixo, onde o cliente tem tempo de buscar a melhor opção dentre os vários serviços disponíveis.

Portanto, é necessário adotar uma abordagem que possibilite o desenvolvimento de aplicações web tornando-as disponíveis, tanto para os dispositivos móveis quanto para os *desktops*, e que considere o tipo de informação solicitado.

A abordagem utilizada para o desenvolvimento de uma aplicação web de acordo com a arquitetura proposta nesta dissertação é diferente do modelo tradicional onde o conteúdo está misturado com a apresentação.

Nesta abordagem o Processador da Camada de Apresentação faz parte da Aplicação Web e é ele quem separa o conteúdo do formato de apresentação e ainda torna possível a disponibilização da Aplicação Web para os diferentes tipos de dispositivos móveis existentes.

5.2.2.1 Processador da Camada de Apresentação

No início da Internet a linguagem HTML era o único formato de apresentação de páginas, e mesmo que cada fabricante de navegador implementasse *tags* específicas, era relativamente fácil de manter as páginas compatíveis com as diversas versões de navegadores. A arquitetura utilizada para o desenvolvimento de uma aplicação web era dividida em três camadas:

- A camada de dados com a funcionalidade de armazenamento e recuperação de dados e que geralmente está associada a um SGBD;
- A camada de negócios que é responsável pela parte lógica da aplicação;
- A camada de apresentação que é a interface com o usuário e está associada aos navegadores web, apresentando as páginas no formato HTML.

Com a evolução da Internet e dos meios de comunicação, chegaram ao mercado novos dispositivos móveis, cada um com características próprias de apresentação de

conteúdo. Surgiram assim novas linguagens de apresentação de conteúdo Web, como por exemplo, WML, cHTML e XHTML.

Para satisfazer a necessidade de se fornecer acesso aos diversos tipos de dispositivos que os clientes estão utilizando para acessar as aplicações web, uma solução seria reescrever a camada de apresentação para cada uma das linguagens de apresentação, de maneira que os diversos formatos de apresentação se comuniquem com a mesma camada de negócios como visto na Figura 21.

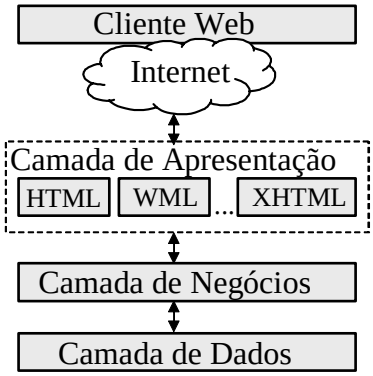

Figura 21: A camada de apresentação dividida em vários formatos

Porém quando se utiliza uma solução deste tipo, quando é necessário fazer algum tipo de mudança, deve-se estendê-la para todos os diferentes formatos de apresentação em separado e o trabalho é, portanto multiplicado.

O Processador da Camada de Apresentação foi incluído na aplicação web para contornar este problema. Ele separa o conteúdo do formato da apresentação, assegurando que o mesmo permaneça independente da apresentação e esta seja flexível e de fácil manutenção, diferente do desenvolvimento de aplicações web em HTML tradicional onde os dados estão misturados com a apresentação. Ele serve para transformar a camada de apresentação de acordo com a linguagem de apresentação do tipo de cliente (dispositivo) que faz a requisição.

O Processador da Camada de Apresentação contém os estilos de cada linguagem, e conforme a solicitação do cliente, ele une o estilo ao conteúdo do banco de dados e mostra o resultado no formato solicitado. Quando é necessário fazer alguma modificação, basta alterar o estilo de apresentação.

Conforme o tipo de dispositivo, as características da aplicação web podem mudar, ou seja, alguns dados podem ser omitidos ou modificados devido ao tamanho de tela e o tipo

de teclado do dispositivo. Através do Processador da Camada de Apresentação é possível criar estilos, escolhendo quais elementos serão visualizados de acordo com o dispositivo escolhido. Na criação dos estilos de apresentação, é possível determinar quais os dados que serão apresentados, qual linguagem de apresentação será utilizada e como estes dados serão apresentados, ou seja, como será a disposição do conteúdo na tela, quais os tamanhos e cores de letras, onde ficarão as figuras, o que será colocado em parágrafos e em tabelas, quais dados serão acessados através de *hiperlinks*, etc.

5.2.3 Servidor Web

O Servidor Web representa o programa que executa o aplicativo responsável por armazenar as páginas web e permitir o acesso de clientes às páginas através de um navegador web.

Este programa opera aceitando requisições HTTP da rede sem fio e enviando a resposta HTTP ao Cliente. A resposta HTTP consiste do documento solicitado pelo cliente através da URL digitada no navegador do dispositivo móvel.

5.2.4 Web

A Web é um sistema de hipertexto distribuído baseado no modelo cliente-servidor. Trata-se de um método para reunir virtualmente vários tipos de informações. Todos os padrões da Web - o protocolo de comunicação HTTP, a linguagem de descrição de páginas HTML e o método de identificação de recursos URL bem como o código-fonte dos programas cliente e servidor – podem ser disponibilizados pela Internet. Além da linguagem HTML, hoje existem outras linguagens de apresentação de conteúdo Web como visto na seção 5.2.2.1.

O protocolo de comunicação utilizado para a conexão à Internet é o TCP/IP, porém a Internet não é o único caminho para disponibilizar conteúdo Web, apesar de ser a mais comum e a mais utilizada. Este conteúdo pode ser disponibilizado, por exemplo, através de uma rede corporativa que não utilize o protocolo IP, ou seja, utilize algum outro protocolo proprietário para acessar conteúdo Web.

5.2.5 Rede sem Fio

Qualquer uma das plataformas de comunicação sem fio citadas no Capítulo 2 pode ser utilizada para o acesso a Web desde que o dispositivo escolhido tenha suporte para a

mesma. O que muda é a mobilidade, que no caso de uma WLAN, equipada com o padrão 802.11, por exemplo, obriga que o dispositivo deva estar dentro da área de cobertura suportada pelo padrão.

Já no caso de plataformas WWAN de segunda ou terceira gerações, ou seja, redes de telefonia digital que transportam voz e dados, (a primeira geração é da telefonia analógica, caracterizada pela transmissão apenas do sinal de voz), a mobilidade é a maior de todas as plataformas e o acesso pode ser feito de qualquer lugar sem restrições de distância desde que haja cobertura.

As plataformas WPAN apresentam a menor mobilidade de todas as plataformas, como no caso do *Bluetooth*, por exemplo, onde a mobilidade se restringe a uma distância de 10m a 100m.

Um ponto importante a ser tratado é de que os dispositivos móveis hoje são multifunção. Nada impede que um *Palm*, por exemplo, tenha suporte a WLAN e possa se conectar à Web através dessa plataforma e, ao mesmo tempo, tenha a função de operar no modo celular utilizando uma plataforma WWAN para conexão à Web.

5.2.6 Cliente

Existe uma grande variedade de dispositivos móveis no mercado: telefones celulares, *PDA's*, *Handhelds*, *Smart Phones*, *Tablet Computers*, *Notebooks*, *Laptops*, etc. A seleção de qual dispositivo vai depender de fatores como: custo, requisitos de mobilidade, parcerias, funções a serem executadas, tamanho de tela e teclado, recursos computacionais, bateria, quanto pesa o aparelho, etc (BRANS, 2002).

Um dos requisitos necessários para a arquitetura proposta quanto ao Cliente é o dispositivo móvel dever possuir um serviço que permita a navegação na Web, uma vez que a aplicação a ser disponibilizada pela arquitetura é uma aplicação web.

5.2.7 Segurança

Questões relacionadas a Segurança são extremamente importantes de serem tratadas, principalmente quando se trata de redes sem fio que possuem uma vulnerabilidade maior em relação a redes cabeadas. Todavia, são demasiado amplas e complexas. Assim sendo, esta questão está fora do escopo deste trabalho, podendo ser tratada em trabalhos futuros.

5.3 Casos de Uso de uma Aplicação Web para Dispositivos Móveis

O modelo proposto pode ser representado pelos casos de uso mostrados na Figura 22, que podem ser especializados de acordo com o cenário, o tipo de aplicação web utilizada e o tipo de cliente que acessa a aplicação.

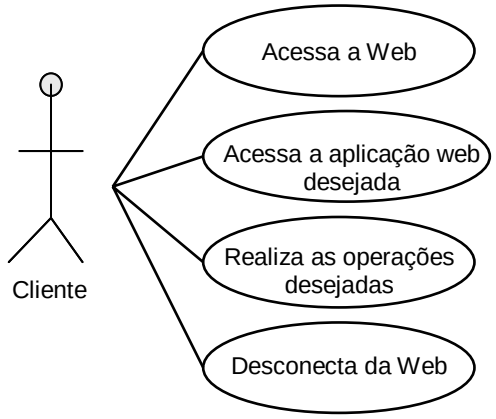

Figura 22: Casos de Uso para uma Aplicação Web para Dispositivos Móveis

No caso de uso onde o ator Cliente acessa a Web, o cliente executa o aplicativo de conexão a rede através do dispositivo móvel e abre o navegador.

Já no caso de uso onde o ator Cliente **Acessa a Aplicação Web Desejada**, este digita ou escolhe o endereço (URL) no navegador do dispositivo móvel para entrar no sítio desejado, enviando uma requisição para o servidor web onde a aplicação está hospedada e recebendo como resposta, o sítio da aplicação web no navegador do dispositivo.

Para o caso de uso onde o ator Cliente **Realiza as Operações Desejadas,** o cliente executa as tarefas de acordo com as opções disponibilizadas pela aplicação web acessada. As tarefas ou ações a serem realizadas dependem do cenário apresentado.

No caso de uso onde o ator Cliente **Desconecta da Web,** este já navegou pela aplicação web e realizou as operações desejadas, portanto fecha o programa navegador e sai do aplicativo de conexão a Web do dispositivo móvel.

Para todos os casos de uso existem outros derivados, como no caso de o cliente não conseguir se conectar a Web, ou a aplicação web não estar disponível no momento do acesso, ou o cliente não conseguir realizar determinada operação, porém estas exceções

não serão consideradas, pois dependem do serviço utilizado para conexão à Web e da aplicação web escolhida pelo cliente.

Vários cenários podem ser citados, como por exemplo:

- Um vendedor acessa o portal da empresa através do celular, verifica a data de entrega de uma ordem de compra de determinado cliente e modifica o dia a ser entregue.
- Um gerente acessa o portal da empresa através de um *palm* e verifica se determinada ordem de produção já foi finalizada e coloca uma observação de urgência na entrega do pedido.
- Um almoxarife acessa o portal da empresa através de um *handheld* e seleciona o relatório de produtos a serem entregues no mês março.
- Um cliente acessa a página de determinado cinema, através de um PDA, e seleciona quais filmes estarão em cartaz do dia 20 ao dia 30 de março de 2006.
- Um motorista verifica, através do celular, como está o trânsito em determinada avenida.
- Um cliente acessa uma página de previsão do tempo através do celular, e escolhe determinada cidade para verificar a previsão do tempo.
- Um cliente acessa a página de um determinado banco através do seu *smartphone*, digita seus dados de número de agência, conta e senha e verifica seu saldo.
- Um cliente acessa a página de uma loja de aparelhos eletrônicos através do *notebook* e faz a compra de uma máquina fotográfica digital com seu cartão de crédito.
- Um supervisor de máquinas recebe uma mensagem pelo celular de que há um problema no chão-de-fábrica e este então desliga uma máquina e invoca uma chamada ao supervisor de manutenção.
- Um diretor tem acesso às informações de CRM por um *palm* e muda prioridades de atendimento de certos clientes críticos.

5.4 Diagrama de Seqüência para a Arquitetura Proposta

Na Figura 23 está mostrado o diagrama de seqüência UML para a arquitetura proposta. Neste diagrama a semântica da UML não é seguida a risca, pois os componentes

do diagrama não são apenas objetos, porém ele serve para mostrar a seqüência em que as ações acontecem e quais os elementos envolvidos.

Seguindo o diagrama, o ciclo começa a partir do Cliente, que pode ser qualquer tipo de dispositivo móvel. Este acessa a Web solicitando uma conexão através de uma plataforma de Rede sem Fio, e faz uma requisição para determinada página do Servidor Web. O Servidor Web recebe esta requisição e como resposta, disponibiliza Aplicação Web escolhida pelo Cliente. O Cliente interage com Aplicação Web através do Servidor seguindo o modelo requisição-resposta, e esta se comunica com o Repositório de Dados e fornece o conteúdo desejado ao Cliente.

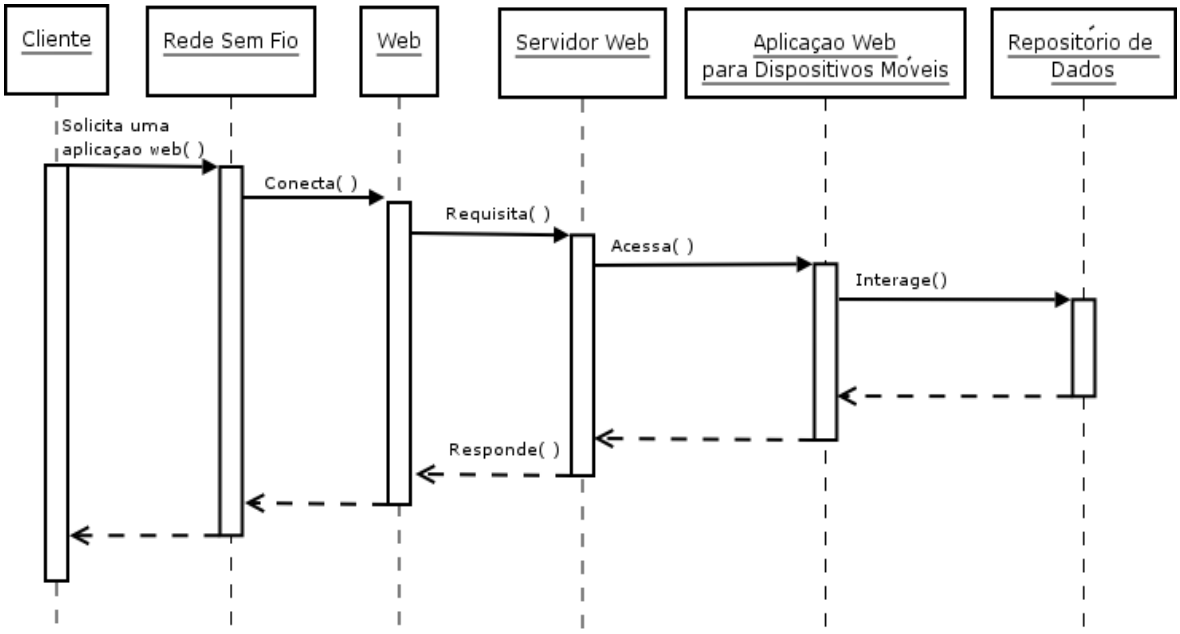

Figura 23: Diagrama de Seqüência para a Arquitetura Proposta

5.5 Conclusões

A arquitetura de software proposta é genérica porque ela é independente de tipo de rede sem fio utilizada pelo dispositivo para o acesso a Web, é independente do protocolo de comunicação utilizado pelo dispositivo e é independente do tipo de dispositivo que acessar a aplicação web disponibilizada.

O que torna a arquitetura genérica é o processador da camada de apresentação, que aplica estilos à mesma de acordo com a linguagem de apresentação solicitada pelo protocolo de acesso a Web suportado pelo dispositivo.

Sendo baseado na linguagem de apresentação do dispositivo, com o processador da camada de apresentação não só é possível disponibilizar uma aplicação web para dispositivos móveis, mas também para todos os dispositivos, inclusive computadores convencionais, nos quais o acesso pode ser feito independente do tipo de rede, seja rede sem fio ou rede cabeada.

A arquitetura propõe uma nova forma de desenvolvimento de aplicações Web, separando o conteúdo da apresentação, garantindo a reusabilidade do código e a simplificação da programação, facilitando igualmente a integração das aplicações web para *e-business* com aplicações web para *m-business*.

6 Implementação e Resultados Experimentais

6.1 Introdução

Este capítulo apresenta os aspectos de implementação da aplicação web para dispositivos móveis na qual o processador da camada de apresentação está incluído, para a validação do modelo conceitual apresentado no capítulo anterior.

Para a implementação do protótipo foram utilizadas as seguintes ferramentas:

- Java Development Kit (JDK) 5.0. A linguagem Java é uma linguagem de programação orientada a objetos desenvolvida pela Sun Microsystems, independente de plataforma (sistema operacional), modular e globalmente aceita como linguagem de programação para aplicações web (JAVA TECHNOLOGY, 2005).
- Ambiente de desenvolvimento Eclipse versão 3.1. Eclipse é uma plataforma desenvolvida em linguagem Java e que opera sob o paradigma de código aberto (*open source*). Possui uma arquitetura de *plug-ins* que permite a integração com outras ferramentas (ECLIPSE, 2005).
- Nokia *Mobile Internet Toolkit* 4.1. É um emulador para dispositivos móveis da Nokia composto por um navegador (*Nokia Mobile Browser*) e por um WAG (*WAP Gateway Simulator*) utilizado nos testes do protótipo desenvolvido (NOKIA, 2005).

A seguir serão apresentados o sistema no qual o protótipo desenvolvido foi validado, o diagrama de classes implementadas, um diagrama de seqüência para uma interação entre cliente e servidor, e, finalmente, os resultados da implementação.

6.2 SC^2

Para a validação do modelo proposto, o Processador da Camada de Apresentação foi desenvolvido como parte da aplicação Web que está sobre o servidor HTTP do sistema SC^2.

O SC^2 (*Supply Chain Smart Coordination*) é um sistema de suporte a decisão que visa oferecer um ambiente integrado para melhor gerenciar cadeias de fornecimento

dinâmicas, cobrindo aspectos de produção, vendas e distribuição (RABELO *et al*., 2002). A arquitetura de implementação do SC[2] pode ser vista na Figura 24.

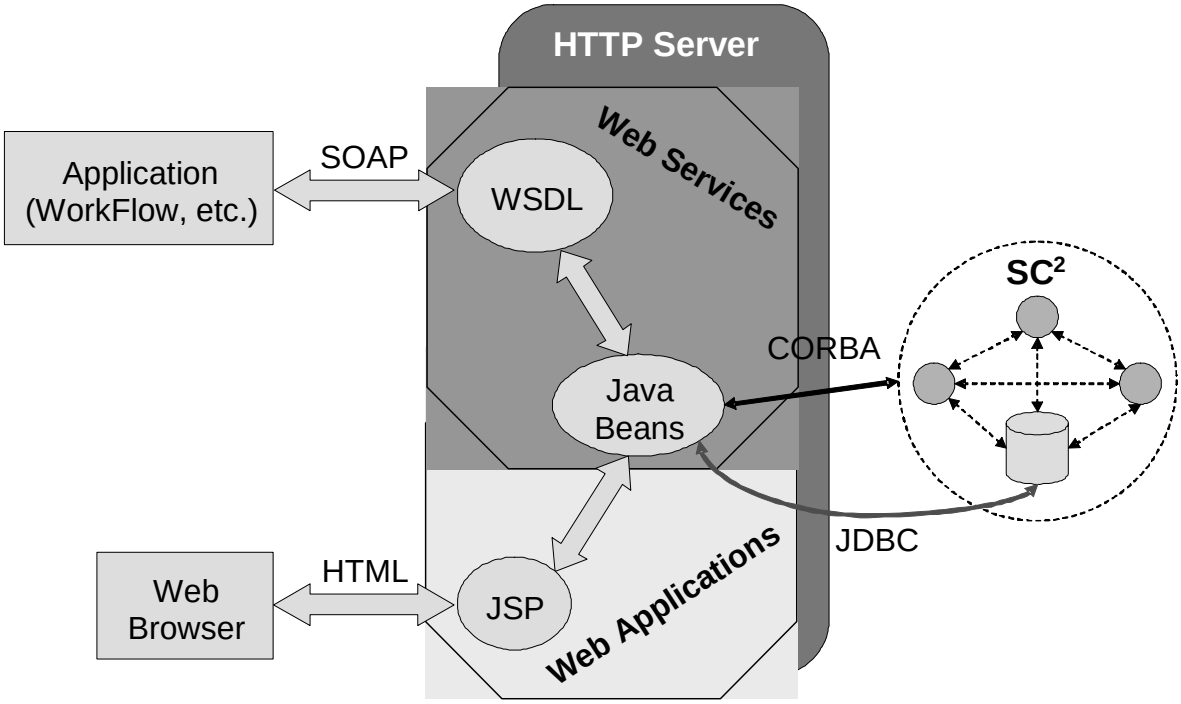

Figura 24: Arquitetura SC[2] (PEREIRA-KLEN *et al.*, 2003).

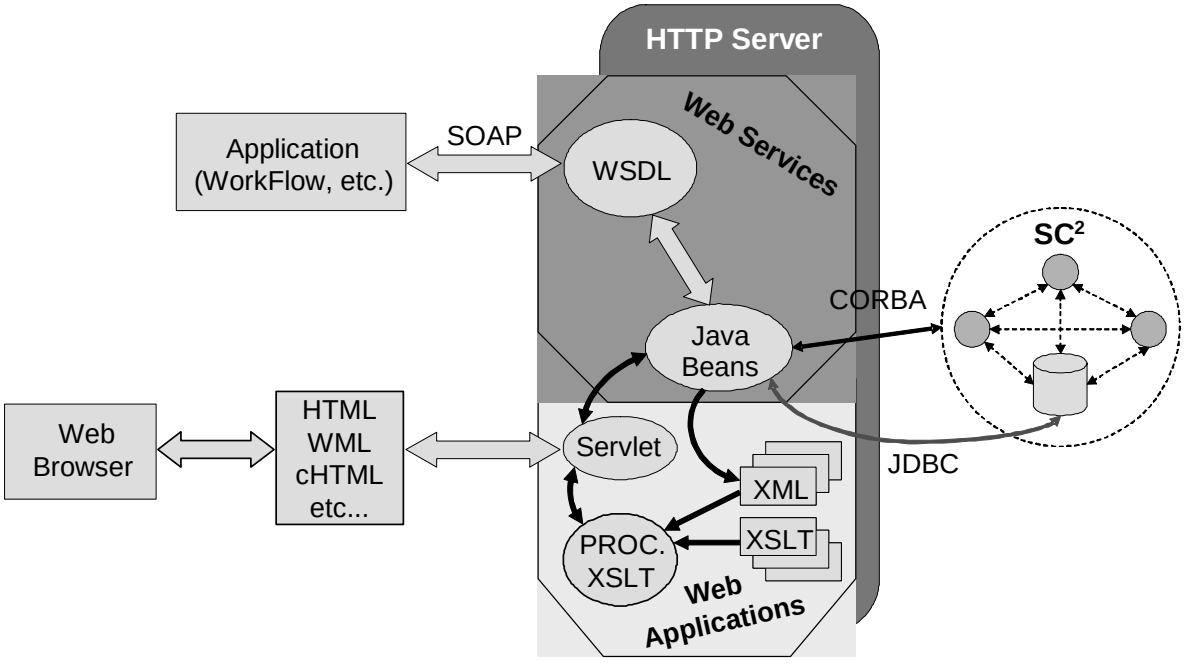

Figura 25: Arquitetura SC[2] modificada. Adaptada de PEREIRA-KLEN *et al.* (2003).

Este sistema possui várias funcionalidades disponíveis na Web como: *Supply Chain Configurator, Distributed Business Process Management, Post-it* e *Ad Hoc Reports*. Maiores detalhes sobre estas funcionalidades podem ser vistos em RABELO *et al*. (2002).

Para os testes de validação da arquitetura genérica proposta nesta dissertação, escolheu-se a funcionalidade *Ad Hoc Reports.* As demais funcionalidades do sistema SC^2 poderiam ser igualmente preparadas para serem acessadas por dispositivos móveis, mas seria apenas um mero trabalho de repetição do que foi feito para aquela funcionalidade escolhida.

Através da funcionalidade *Ad Hoc Reports* são disponibilizadas na Web, informações referentes a ordens de compras, de produção, de entrega, etc. das cadeias de fornecimento. Atualmente estas informações são disponibilizadas pela aplicação web através de JSP e apenas em HTML, conforme a Figura 24.

A modificação feita na arquitetura do SC^2 na qual o Processador da Camada de Apresentação, agora chamado Processador XSLT é incluído na aplicação web para disponibilizar as informações fornecidas pela funcionalidade *Ad Hoc Reports* pode ser vista na Figura 25. A seguir serão explicadas as principais mudanças na arquitetura do SC^2.

6.2.1 XML E XSLT

O padrão XSLT foi escolhido para o desenvolvimento do Processador da Camada de Apresentação pelo fato de ser um padrão e dar suporte ao modelo conceitual proposto no Capítulo 5.

Os objetos do banco de dados (*Java Beans*) utilizados pela funcionalidade *Ad Hoc Reports* do sistema SC^2 foram convertidos para documentos **XML** para a utilização do padrão XSLT e *templates* (***XSL Stylesheets***) foram criados para as linguagens de apresentação desejadas.

As linguagens WML e XHTML foram escolhidas para validar a arquitetura proposta pela quantidade de dispositivos e emuladores com suporte a essas linguagens. É importante afirmar que, independente do dispositivo a ser escolhido para mostrar a aplicação web, o que interessa para o Processador XSLT é a linguagem de apresentação suportada pelo mesmo, não importando se é um celular da segunda geração ou um *palm* de terceira geração. É baseado nessa linguagem que o Processador XSLT escolhe o estilo e faz a transformação.

Quanto à criação dos *templates* é necessário saber para que tipos de dispositivos e quais os objetivos da aplicação web a ser disponibilizada, para que a mesma seja adequada ao tipo de dispositivo que irá acessá-la. Alguns *templates* criados para a validação do protótipo podem ser visualizados no Anexo A apenas para mostrar como estes foram desenvolvidos para a cada linguagem de apresentação.

6.2.2 Servlets

Uma das modificações na arquitetura do SC^2 é a utilização de *Servlets* para o controle da aplicação web. Isto porque agora o conteúdo não será disponibilizado apenas em HTML (como era com o JSP) e a tecnologia de desenvolvimento da aplicação é baseada em XSLT.

Neste caso, o processo de transformação ocorre dentro de um s*ervlet* que recebe uma requisição e direciona o cliente para a página desejada de acordo com a linguagem de apresentação suportada pelo dispositivo do mesmo.

6.2.3 Processador XSLT

O processador XSLT está dentro de um s*ervlet* é obtido através da classe Java chamada **TransformerFactory** do pacote **javax.xml.transform**. O objeto **factory** dessa classe retorna a implementação do processador XSLT, que implementa a interface **Transformer**. Na criação do processador XSLT, o **factory** recebe como argumento um arquivo XSL (*template*) que será utilizado pelo processador XSLT no momento da transformação.

No processo de transformação o processador XSLT recebe como parâmetros um fluxo de entrada e um fluxo de saída. O fluxo de entrada contém o documento XML e o fluxo de saída, foi direcionado, nessa implementação, para o fluxo de saída da resposta HTTP do *servlet* obtendo o *stylesheet* da linguagem desejada, ou seja, a linguagem de apresentação do dispositivo do cliente.

6.2.4 MIME Types

Na implementação da aplicação web, para diferenciar o tipo de cliente que está fazendo a requisição, são utilizados MIME *(Multipurpose Internet Mail Extensions*) *types*. O MIME *type* é uma especificação desenvolvida pelo IETF (*Internet Engineering Task Force*) que define mecanismos para enviar outros tipos da informação no *E-mail* usando

caracteres de codificação diferentes do ASCII (*American Standard Code for Information Interchange*), incluindo texto, arquivos com imagens, sons, vídeos e programas de computador. Cada tipo de arquivo tem o seu *MIME type* específico.

O MIME é também um componente fundamental de protocolos de comunicação como o HTTP, que requer que os dados sejam transmitidos no contexto de mensagens de *e-mail*, mesmo que os dados não sejam um *e-mail* propriamente dito.

O MIME *type* é constituído de duas partes, o tipo principal do arquivo e o subtipo, e é representado escrevendo-se o tipo e o subtipo separados por uma barra "/". Por exemplo: text/html, image/gif, video/mpeg, video/quicktime.

Na aplicação web desenvolvida há *servlets* que testam o MIME *type* que está no cabeçalho da requisição HTTP feita pelo cliente e se baseiam nesse MIME para direcionar o cliente para a página desejada. Os MIME *types* utilizados para a aplicação desenvolvida podem ser vistos na Figura 26.

Conteúdo	MIME *type*
WML	text/vnd.wap.wml
XHTML	application/xhtml+xml

Figura 26: MIME *types* para aplicações nas linguagens WML e XHTML.

6.3 Diagramas de Classes Implementadas

Foram implementados dois pacotes de classes para possibilitar a disponibilização da aplicação web para as várias linguagens de apresentação existentes. Os pacotes são **br.ufsc.gsigma.webFramework** e **br.ufsc.gsigma.sc2.webFramework** e podem ser visualizados nas Figuras 27 e 28.

O pacote **br.ufsc.gsigma.webFramework** é a parte genérica da aplicação web e pode ser visto como um *framework*. Isto porque é um conjunto de classes e interfaces que cooperam para resolver um tipo de problema de software, e neste caso, cooperam para disponibilizar a aplicação web para as várias linguagens de apresentação existentes. Esse pacote possui uma infra-estrutura reutilizável e um comportamento genérico que pode ser adaptado para várias aplicações.

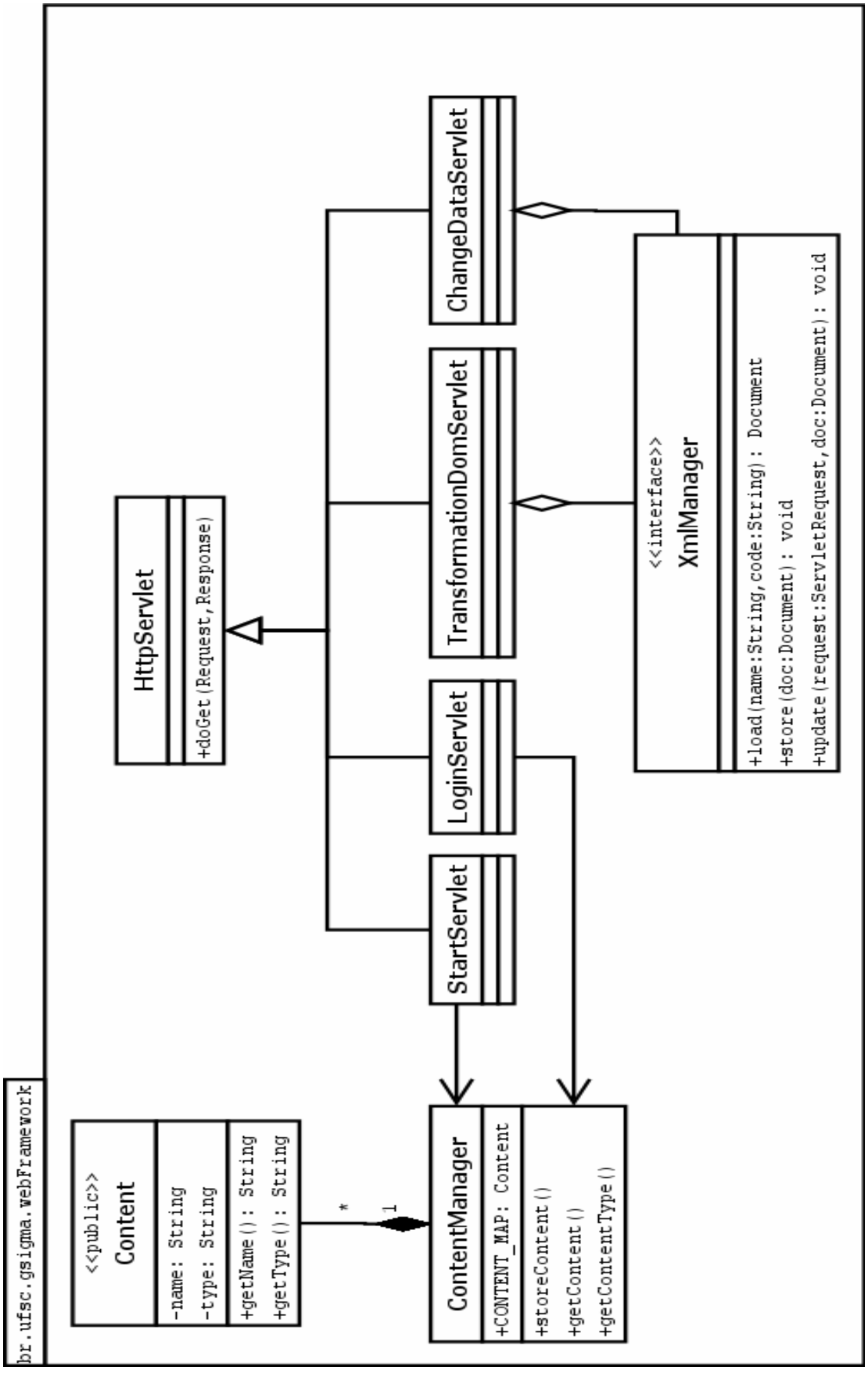

Figura 27: Diagrama de Classes do pacote **br.ufsc.gsigma.webFramework**

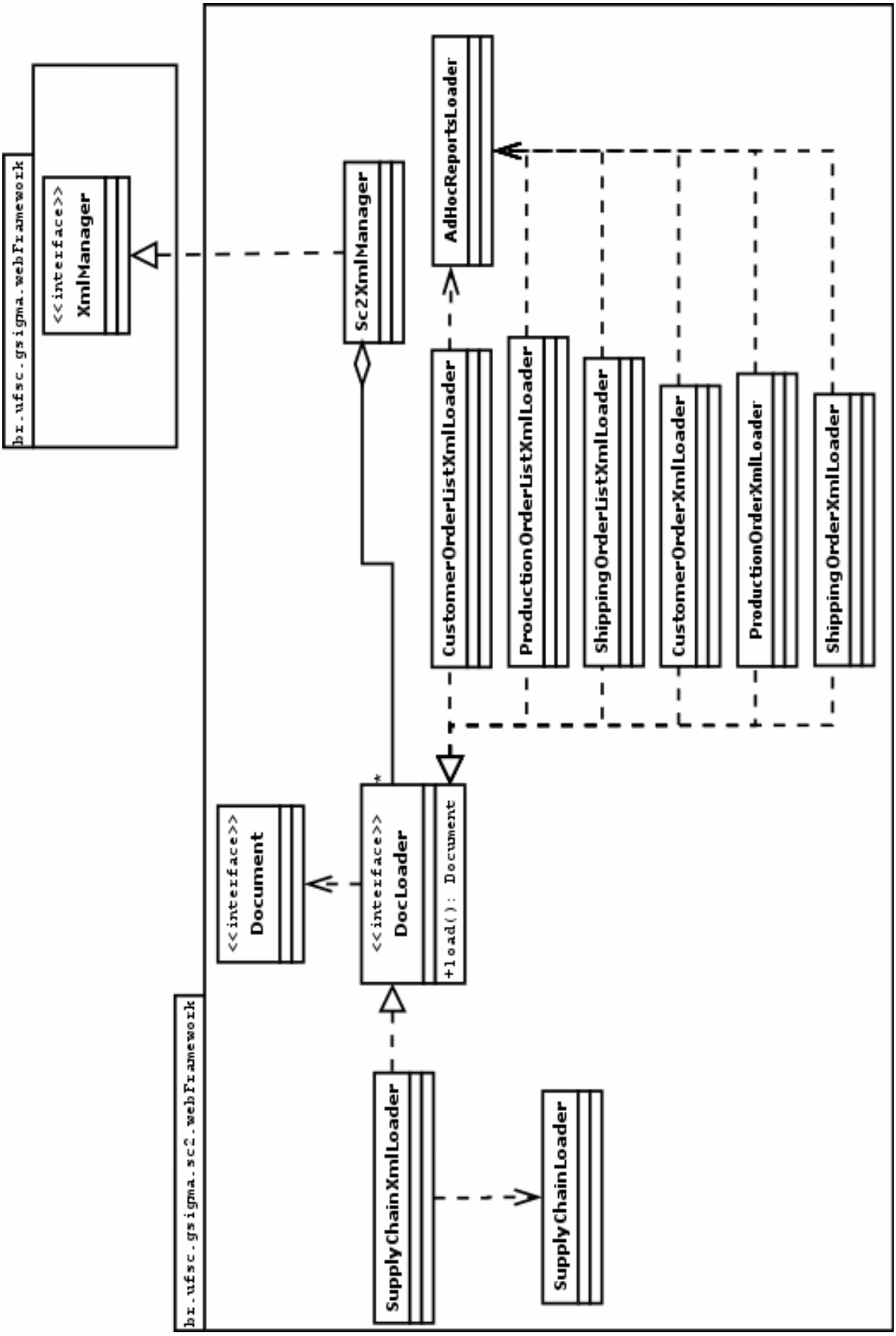

Figura 28: Diagrama de Classes do pacote **br.ufsc.gsigma.sc2.webFramework**

No pacote **br.ufsc.gsigma.sc2.webFramework** estão todas as classes desenvolvidas para que o sistema do SC^2 possa fornecer os dados solicitados pela aplicação web no formato XML para que possam ser utilizados pelo Processador XSLT no momento da transformação e aplicação do estilo solicitado na requisição HTTP feita pelo cliente.

6.3.1 Pacote br.ufsc.gsigma.webFramework

Nesta seção serão explicadas as classes que fazem parte do pacote **br.ufsc.gsigma.webFramework** para melhor entendimento do diagrama de classes da Figura 27.

A classe (ou o *Servlet*) **StartServlet** testa o MIME *type* que está no cabeçalho da requisição HTTP feita pelo cliente e se baseia nesse MIME para direcionar o cliente para a página desejada. Ela utiliza, ou depende das classes auxiliares **Content** e **ContentManager** que capturam e armazenam o tipo conteúdo da requisição HTTP feita pelo cliente.

A classe (ou o *Servlet*) **LoginServlet** além de capturar o tipo de conteúdo da requisição HTTP, também valida as informações de *Login* e *Password* de acordo com as informações do formulário HTTP da página de entrada do Sistema fornecidas pelo cliente.

Na classe (ou no *Servlet*) **TransformationDomServlet** o Processador XSLT é criado, e todo o processo de transformação ocorre conforme explicado na seção 5.3.3 e o documento XML do fluxo de entrada é carregado através da interface **XmlManager**.

A interface **XmlManager** é a ponte entre o *framework* genérico (pacote **br.ufsc.gsigma.webFramework)** e o sistema legado do SC^2 e é responsável por carregar, armazenar e atualizar um documento XML gerado através das classes criadas no pacote **br.ufsc.gsigma.sc2.webFramework.**

Através da classe **ChangeDataServlet** é permitido ao usuário alterar dados de um determinado documento de acordo com a aplicação em questão. No caso do sistema SC^2 é permitido ao usuário alterar o *STATUS* de uma ordem de compra ou de uma ordem de entrega de determinada cadeia de suprimentos.

O código das classes implementadas no pacote **br.ufsc.gsigma.webFramework** pode ser visto no anexo B.1.

6.3.2 Pacote br.ufsc.gsigma.sc2.webFramework

Nesta seção serão igualmente explicadas as classes que fazem parte do pacote **br.ufsc.gsigma.sc2.webFramework** para melhor entendimento do diagrama de classes da Figura 28.

As classes **CustomerOrderListXmlLoader**, **ProductionOrderListXmlLoader**, e **ShippingOrderListXmlLoader**, e as classes **CustomerOrderXmlLoader**, **ProductionOrderXmlLoader** e **ShippingOrderXmlLoader** implementam a interface **DocLoader** e dependem da classe **AdHocReportsLoader** do sistema legado SC^2.

A classe **SupplyChainXmlLoader** também implementa a interface **DocLoader** e depende da classe **SupplyChainLoader** do sistema legado SC^2.

A interface **DocLoader** é responsável por carregar um arquivo XML específico baseado em um código. Esse código pode ser de o código de determinada cadeia de suprimento do sistema legado SC^2, por exemplo.

A classe **Sc2XmlManager** implementa a interface **XmlManager** do pacote **br.ufsc.gsigma.webFramework** e administra a lista de documentos a serem carregados armazenados e modificados.

Os pacotes de classes **br.ufsc.gsigma.sc2.beans**, **br.ufsc.gsigma.sc2.beans.adhoc** e **br.ufsc.gsigma.sc2.db** do sistema legado SC^2 também foram utilizados na implementação do pacote **br.ufsc.gsigma.sc2.webFramework.**

O código de algumas classes implementadas e que são relevantes de serem mostradas para ilustrar o desenvolvimento do pacote **br.ufsc.gsigma..sc2.webFramework** pode ser visto no anexo B.2.

6.4 Diagrama de Seqüência – Interação entre Cliente e Servidor

Na Figura 29 está o diagrama de seqüência dos objetos e métodos envolvidos na implementação das classes do protótipo de software que representam as interações entre o lado cliente e o lado servidor da aplicação web.

Neste diagrama estão mostrados todos os métodos e objetos envolvidos a partir do momento que um determinado cliente faz uma requisição HTTP para o servidor web e finaliza com a resposta do servidor para o cliente. Este processo ocorre da mesma forma a cada requisição HTTP feita pelo cliente.

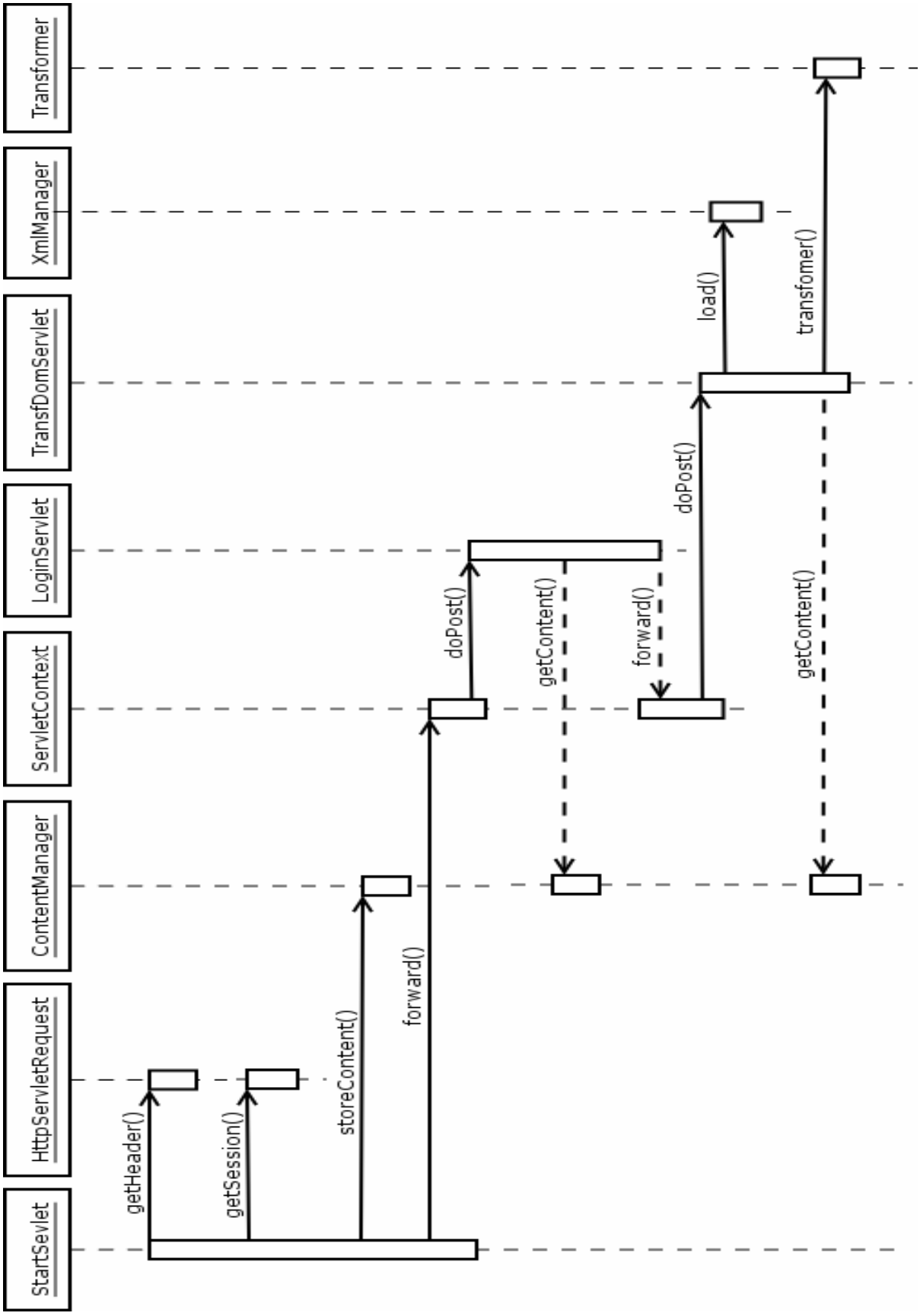

Figura 29: Diagrama de Seqüência –Interação entre Cliente e Servidor

A interação entre os objetos também pode ser visualizada analisando-se o código das classes implementadas que está no Anexo B desta dissertação.

6.5 Recomendações para a Implantação da Arquitetura Genérica de Software

Após apresentar as tecnologias utilizadas para o desenvolvimento da aplicação web, já aplicadas ao caso específico do sistema SC^2, pode-se chegar a determinadas conclusões que servem de guia para a implantação da arquitetura genérica de software proposta em empresas com sistemas legados que optarem por esta solução, e em empresas que pretendam iniciar o desenvolvimento de aplicações web baseadas nessa arquitetura.

6.5.1 Implantação da Arquitetura Genérica de Software em Sistemas Legados

Para a implantação da arquitetura de software proposta em sistemas legados é necessário avaliar as implicações que a mudança na forma de desenvolvimento de aplicações web já desenvolvidas pode ocasionar, pois não é tão fácil modificar a cultura de desenvolvimento das aplicações web da empresa.

Partindo para questões práticas o que precisa ser feito é:

- O ambiente de desenvolvimento da aplicação web vai depender da linguagem de programação utilizada.
- A linguagem de programação utilizada deve ter suporte ao padrão XSLT que é a tecnologia de desenvolvimento do processador da camada de apresentação e toda arquitetura esta firmada sobre este padrão.
- Caso a linguagem de programação Java já esteja sendo utilizada nos sistema legado, o *framework* exposto na seção 6.3.1 pode ser utilizado e pacote **br.ufsc.gsigma.webframework** pode ser facilmente adaptado ao contexto da aplicação.
- Quanto ao banco de dados basta, que de alguma maneira, os dados sejam convertidos para XML para a utilização do padrão XSLT. Isto pode ser feito com recursos da linguagem de programação utilizada ou com ferramentas apropriadas;

- Verificar se as tecnologias utilizadas para o desenvolvimento de aplicações web atualmente são compatíveis ou podem ser adaptadas para funcionar nos moldes da arquitetura proposta, senão terão que ser substituídas.
- A criação dos estilos de apresentação (*templates*) deve ser feita de acordo com os tipos de dispositivos móveis que se deseja atingir e com os objetivos específicos de cada aplicação.

6.5.2 Implantação da Arquitetura Genérica de Software em Sistemas a serem Desenvolvidos

Em sistemas a serem desenvolvidos, a implantação da arquitetura genérica de software para o desenvolvimento de aplicações web se torna mais fácil, pois o processo é iniciado do zero e a reengenharia não é necessária, basta que sejam contratados desenvolvedores especializados nas tecnologias envolvidas na aplicação ou até mesmo treinar os desenvolvedores para modificar a cultura de desenvolvimento das aplicações web da empresa.

Neste caso, considerando novamente as questões práticas:

- O ambiente de desenvolvimento da aplicação web vai depender da linguagem de programação utilizada.
- O *framework* exposto na seção 6.3.1 pode ser utilizado caso a linguagem Java seja escolhida para o desenvolvimento e cada classe do pacote **br.ufsc.gsigma.webframework** pode ser facilmente adaptada ao contexto da aplicação escolhida.
- A linguagem de programação utilizada deve ter suporte ao padrão XSLT que é a tecnologia de desenvolvimento do processador da camada de apresentação e toda arquitetura esta firmada sobre este padrão.
- Como o desenvolvimento está sendo iniciado do zero, o ideal é utilizar uma ferramenta que já disponibilize o conteúdo do banco de dados em XML.
- A criação dos estilos de apresentação (*templates*) deve ser feita de acordo com o tipo de dispositivos móveis que se deseja atingir e com os objetivos específicos de cada aplicação.

6.6 Resultados Experimentais

Nesta seção será feita uma demonstração do sistema mostrando a seqüência de telas do protótipo desenvolvido para a validação. Os *templates* foram desenvolvidos para as linguagens WML e XHTML pela quantidade de emuladores e dispositivos móveis com suporte a essas linguagens. As telas mostradas são do emulador para dispositivos móveis da Nokia (NOKIA, 2005). Esse emulador tem suporte a uma variedade de dispositivos móveis da Nokia e as figuras mostradas emulam um celular, para a linguagem WML e um *palm*, para a linguagem XHTML, respectivamente. As informações mostradas são referentes à ordens de produção e de entrega de cadeias de suprimento do sistema SC^2.

6.6.1 Seqüência de Passos

Entrar na Internet pelo dispositivo móvel.

Digitar a URL: http://www.gsigma.ufsc.br/sc2wap/.

Digitar como Login: user e Password: user, como na Figura 30.

Escolher **Options → Submit → Select** para a versão WML. Para a versão XHTML basta escolher o botão **Submit.**

.a) b)

Figura 30: Login SC^2 WML (a) e Login SC^2 XHTML (b)

Escolher a cadeia de suprimento desejada (Figura 31). Como exemplo, a Cadeia de Suprimento SC1 foi escolhida.

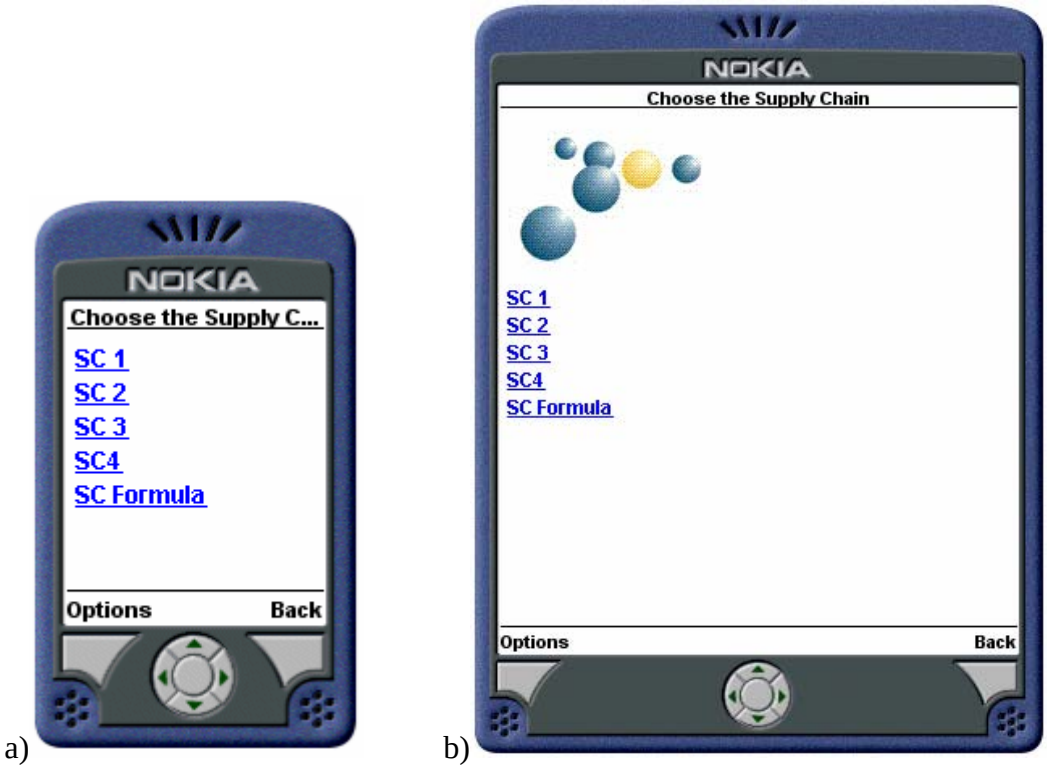

Figura 31: Escolha da Cadeia de Suprimento WML (a) e XHTML (b).

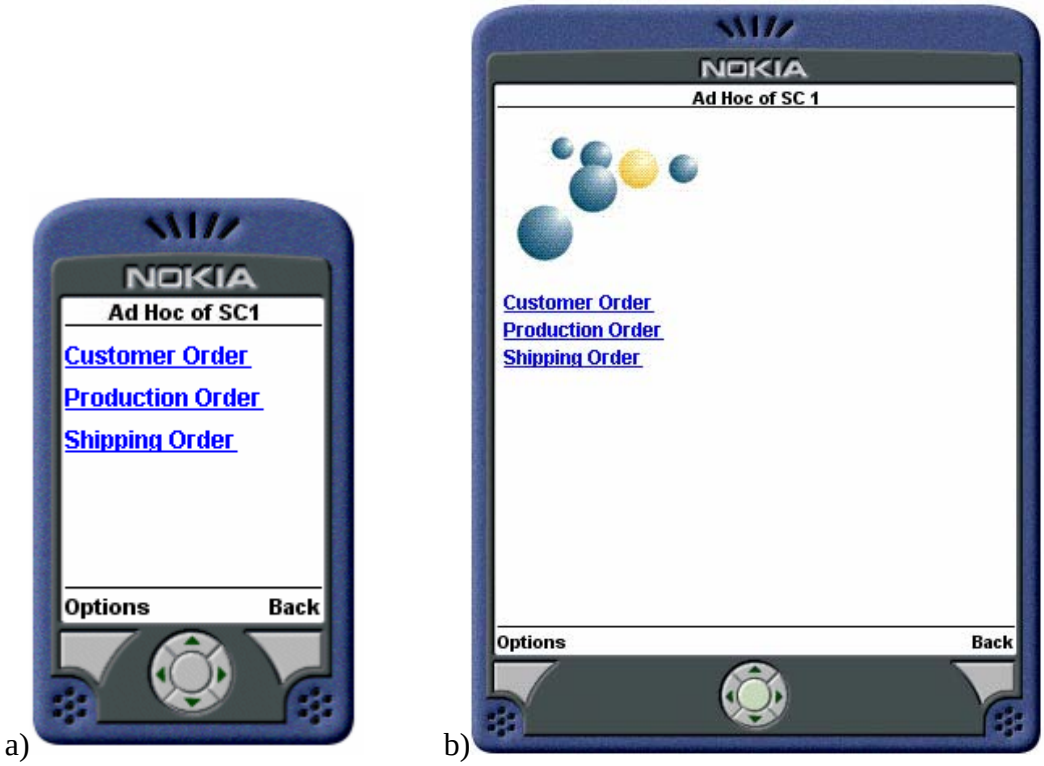

Figura 32: Escolha do Relatório WML (a) e XHTML (b).

Escolher o relatório desejado (Figura 32).

a) b)

Figura 33: Lista de ordens de entrega da Cadeia de Suprimento SC1 WML (a) e Lista de ordens de produção da Cadeia de Suprimento SC1 XHTML (b).

a) b)

Figura 34: Ordem de entrega da Cadeia de Suprimento SC1 WML (a) e Ordem de produção da Cadeia de Suprimento SC1 XHTML (b).

Foram escolhidos, o relatório de ordens de entrega para a versão WML e o relatório de ordens de produção para a versão XHTML (Figura 33).

Na Figura 34 podem ser visualizados, o conteúdo da ordem de entrega **SO1** na versão WML e o conteúdo da ordem de produção **P1** na versão XHTML. Escolhendo **Options**→ Item 1 → **Select**, pode se visualizar detalhes a respeito dos itens da ordem de entrega **SO1** na versão WML. Na versão XHTML os detalhes a respeito dos itens da ordem de produção **P1** já estão mostrados na tela principal.

Na Figura 35 pode ser vista a opção de mudança de STATUS para uma ordem de entrega na versão WML e para uma ordem de produção na versão XHTML. Através dessa opção o STATUS de determinada ordem, seja de produção ou de entrega, pode ser alterado. Essa opção pode ser utilizada em vários setores, como por exemplo, um operador do setor de produção pode alterar o STATUS de uma ordem de produção de "IN PROGRESS" para "COMPLETED", quando a ordem é finalizada.

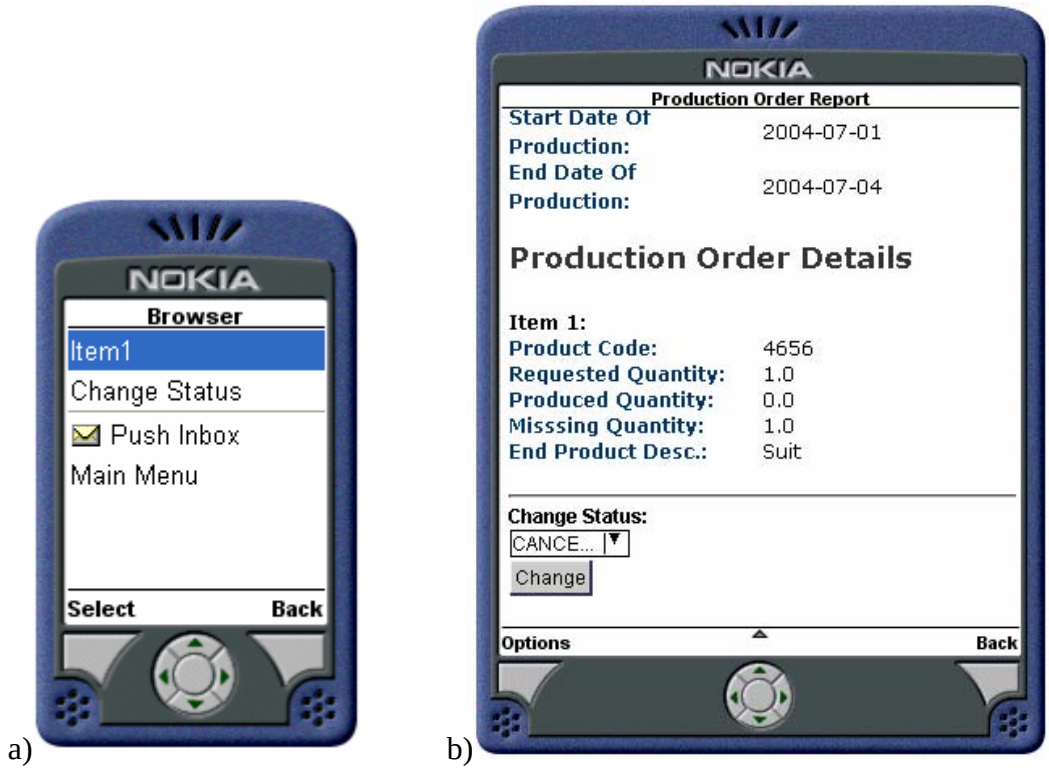

Figura 35: Opção de mudança de STATUS WML (a) e XHML (b)

Na Figura 36 podem ser vistas as opções de mudança de STATUS para as ordens de entrega e de produção nas versões WML e XHTML, respectivamente, finalizando a demonstração da navegação pelo sistema implementado.

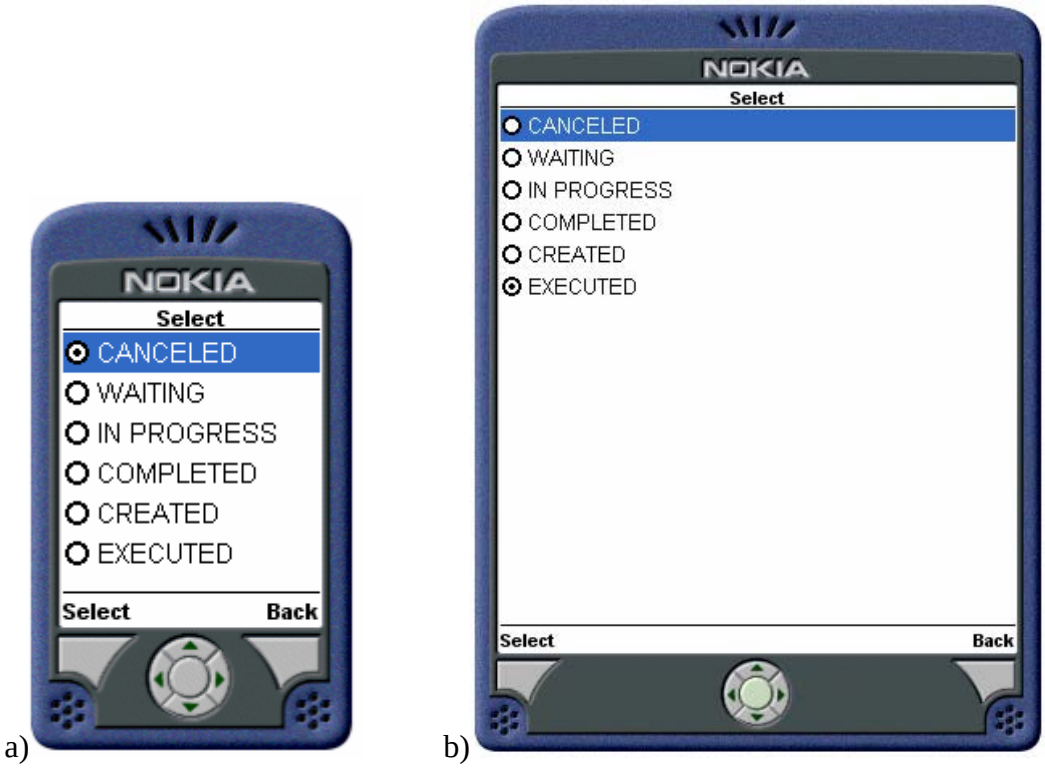

Figura 36: Escolha do novo STATUS WML (a) e XHTML (b).

6.6.2 Dispositivos e Emuladores Utilizados

Além do emulador para dispositivos móveis da Nokia, outros emuladores foram utilizados nos testes do protótipo desenvolvido, como TagTag Emulator (TT, 2005) disponível apenas para a linguagem WML, Openwave Phone Simulator (OPENWAVE, 2005) e WAP Proof (WAP PROFF, 2005), ambos disponíveis para WML e XHTML. Todos funcionaram de maneira adequada, com exceção de alguns *bugs* no emulador WAP Proof que às vezes não mostrava o conteúdo adequadamente.

O emulador da Nokia foi utilizado porque abrange uma variedade de dispositivos móveis da Nokia e não apenas alguns modelos de celular, como o emulador WAP Proof, por exemplo, e tem suporte as linguagens WML e XHTML.

Também foram realizados testes com dispositivos móveis reais, como: celulares Siemens A57, Nokia 3105, SonyEricsson T630, Samsung E700 e Motorola V300 e um *palm* Palm® TX, e todos apresentaram os resultados de acordo com os testes feitos nos emuladores. Com base em todos esses testes nesses dispositivos e emuladores afirma-se que a solução desenvolvida é genérica.

6.7 Conclusões da Implementação

Após a implementação pode-se concluir que:

- O acesso às informações disponibilizadas pode ser feito de qualquer dispositivo móvel que possua um navegador WML ou XHTML independente de tecnologia de desenvolvimento, provando que a disponibilização da Aplicação Web para Dispositivos Móveis, de acordo com a arquitetura apresentada, é independente do tipo de dispositivo móvel, do sistema operacional (tanto dos servidores como dos dispositivos), do protocolo de comunicação e da arquitetura de rede sem fio utilizada.

- Através da arquitetura é possível disponibilizar aplicações web não apenas para dispositivos móveis, mas também para *desktops* conectados a redes cabeadas ou redes sem fio, pois ela é independente de dispositivo e de rede.

- A visualização do conteúdo vai depender do tamanho da tela do dispositivo, isto é, dependendo da quantidade de conteúdo, barras de rolagem podem ser utilizadas e a disposição do conteúdo na tela pode variar de acordo com o aplicativo navegador utilizado.

- Para cada linguagem de apresentação, devem ser criados novos *templates* de acordo com a aplicação em questão. Nesta implementação foram criados *templates* para as linguagens WML e XHTML, mas nada impede que sejam criados *templates* para outras linguagens de apresentação.

- Outra questão é quanto a criação dos *templates* de acordo com determinada classe de dispositivos a serem atingidos. Na prática, o mesmo conteúdo que é mostrado numa tela de um *desktop*, não é adequado para ser mostrado na tela de um *palm*, por exemplo. O que se costuma querer mostrar num dispositivo móvel são apenas algumas informações que a aplicação web é capaz de fornecer. Logo, é necessário definir o que e como se deseja mostrar.

7 Conclusões

Este trabalho propôs uma arquitetura genérica de software para a disponibilização de aplicações web para Computação Móvel que dá suporte aos vários tipos de dispositivos móveis existentes, visto que no mercado atual a solução fornecida geralmente está atrelada à determinada marca, tipo, protocolo de comunicação e sistema operacional do dispositivo em questão. O desenvolvimento de uma aplicação web de acordo com a arquitetura proposta utiliza uma abordagem baseada em padrões abertos, facilitando o desenvolvimento e tornando a solução reutilizável.

A OMA, uma organização empenhada em assegurar a interoperabilidade entre dispositivos, trabalha para unificar todas as linguagens de apresentação existentes. Porém, enquanto isso não acontece, a diversidade de linguagens e tipos de dispositivos, desde os mais antigos, até os mais modernos, ainda existe. O desenvolvimento da aplicação web para dispositivos móveis leva em consideração estes fatores, pois é o cliente que decide qual dispositivo quer utilizar e tudo deve ser adequado a esta escolha.

Quanto a utilização de redes sem fio, esta pode ser vista como uma outra opção de conexão; desta forma a inserção desta tecnologia não impacta o funcionamento do restante da infra-estrutura de tecnologia de informação. Ela é uma nova opção de conectividade que deve adicionar grande valor ao negócio da empresa que desenvolver e implantar aplicações específicas.

Além da arquitetura, pode-se afirmar que um *framework* para o desenvolvimento de aplicações web foi concebido e este pode ser utilizado numa variedade de aplicações web nas quais o padrão XSLT é adotado como forma de apresentação de conteúdo através da utilização de *templates* específicos.

A aplicação web desenvolvida e o código-fonte do *framework* de desenvolvimento estão disponíveis no sítio do GSIGMA (Grupo de Sistemas Inteligentes de Manufatura) na área de protótipos de software sob o ícone SC^2 *WIRELESS* (GSIGMA, 2006).

7.1 Trabalhos Futuros

Algumas questões não foram tratadas nesse trabalho devido às suas complexidades. No entanto, são muito importantes e darão continuidade na pesquisa para o desenvolvimento de aplicações Web envolvendo redes sem fio e dispositivos móveis.

A seguir são citadas algumas sugestões para trabalhos futuros:

- **Tratar de questões relacionadas à segurança, autenticação e criptografia**

Segurança, autenticação e criptografia são questões muito importantes quando se trata de redes sem fio que possuem uma vulnerabilidade maior em relação a redes cabeadas.

Algum mecanismo de segurança deve ser incluído para que somente usuários autorizados tenham acesso ao dispositivo móvel e informações confidenciais devem ser encriptadas. E para construir mecanismos de segurança eficientes é necessário analisar as vulnerabilidades, o valor dos dados que estão sendo protegidos e a motivação dos invasores em potencial.

No Projeto WILMA há um estudo que leva em consideração protocolos e meios de acesso sem fio e envolve várias tecnologias de desenvolvimento dando ênfase para mecanismos de autenticação, segurança, QoS, etc, utilizados na troca de informações (WILMA, 2001).

- **Tornar a aplicação global.**

No mundo globalizado a criação de aplicações globais que são desenvolvidas independente dos países ou linguagem dos usuários e ajustadas para os requisitos dos vários países ou regiões é muito importante no caso da empresa possuir filiais em vários países e em cada um deles é falado um idioma diferente. Um exemplo pode ser visto no trabalho de (DABKOWSKI *et al.*, 2003).

- **Pesquisar a utilização do padrão DISelect (*Content Selection for Device Independence*)**

O padrão DISelect está sendo desenvolvido pelo DIWG (*Device Independence Working Group*) e é baseado num perfil independente de dispositivo para a linguagem XHTML (W3C, 2004). Esta especificação representa uma parte da pesquisa desenvolvida pelo DIWG para a provisão de uma linguagem de marcação para a criação de aplicações web que podem ser usadas numa variedade de dispositivos com características diferentes.

A meta é sugerir que o conteúdo de aplicações web possa ser criado, gerado ou adaptado para uma melhor experiência do usuário quando disponibilizado em diferentes

meios de acesso. Estes meios podem incluir uma diversidade de dispositivos, agentes de usuário, canais, modalidades, formatos etc.

Anexo A - *Templates* XSL Desenvolvidos para a Funcionalidade *Ad Hoc Reports* do Sistema SC^2

A.1 – *Templates* XSL para a linguagem WML

1) Template XSL para a Lista de Cadeia de Suprimentos do Sistema SC^2

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output doctype-public="-//WAPFORUM//DTD WML 1.1//EN" doctype-
system="http://www.wapforum.org/DTD/wml_1.1.xml" indent="yes"/>

<xsl:template match="/">
<wml>

<card title="Choose the Supply Chain">
<p>
<xsl:for-each select="allSupplyChains/supplyChain">
<xsl:sort select="." />
<a>
<xsl:attribute name="href">wml/Order.jsp?code=<xsl:value-of
select="@code" />&amp;name=<xsl:value-of select="." />
</xsl:attribute>
<xsl:value-of select="." />
</a>
<br />
</xsl:for-each>
</p>
</card>
</wml>
</xsl:template>

</xsl:stylesheet>
```

2) Template XSL para a lista de Ordens de Compra do Sistema SC^2

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output doctype-public="-//WAPFORUM//DTD WML 1.1//EN" doctype-
system="http://www.wapforum.org/DTD/wml_1.1.xml" indent="yes"/>

<xsl:template match="/">
<wml>
<card title="Choose the Customer Order">
```

```
<p>
<xsl:for-each select="allCustomerOrders/customerOrder">
<xsl:sort select="." />
<a>
<xsl:attribute
name="href">/sc2wap/TransformationDomServlet?name=Customer_Order&amp;code
=<xsl:value-of select="/allCustomerOrders/@scCode" />|<xsl:value-of
select="." />
</xsl:attribute>
<xsl:value-of select="." />
</a>
<br />
</xsl:for-each>
</p>
</card>
</wml>
</xsl:template>

</xsl:stylesheet>
```

3) Template XSL para uma Ordem de Compra do Sistema SC²

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:output doctype-public="-//WAPFORUM//DTD WML 1.1//EN" doctype-
system="http://www.wapforum.org/DTD/wml_1.1.xml" indent="yes"/>

<xsl:template match="/">
    <wml>
    <xsl:apply-templates/>
    </wml>
</xsl:template>

<xsl:template match="CustomerOrder">
<card title="Customer Order">
<p>
<b><small>Order Number: </small></b>
<br/>
<b> <xsl:value-of select="OrderNumber"/> </b>
<br/>
<b><small>Order Date:</small></b>
<br/>
<b> <xsl:value-of select="OrderDate"/> </b>
<br/>
<b><small>Delivey Date:</small> </b>
<br/>
<b> <xsl:value-of select="DeliveryDate"/></b>
<br/>
<b><small>Client Code:</small> </b>
<br/>
<b> <xsl:value-of select="ClientCode"/></b>
<br/>
<b><small>Client Name:</small> </b>
<br/>
<b> <xsl:value-of select="ClientName"/></b>
```

```
<br/>
<b><small>Sales Agent:</small> </b>
<br/>
<b> <xsl:value-of select="SalesAgent"/></b>
<br/>
</p>
<xsl:for-each select="CODetail/item">
<do type="accept">
<xsl:attribute name="name">do<xsl:value-of
select="Numbering"/></xsl:attribute>
<xsl:attribute name="label">Item <xsl:value-of
select="Numbering"/></xsl:attribute>
<go>
<xsl:attribute name="href">#item<xsl:value-of
select="Numbering"/></xsl:attribute>
</go>
</do>
</xsl:for-each>
</card>
<xsl:apply-templates select="CODetail"/>
</xsl:template>

<xsl:template match="CODetail">
<xsl:for-each select="item">
<card>
<xsl:attribute name="id">item<xsl:value-of
select="Numbering"/></xsl:attribute>
<xsl:attribute name="title">Item <xsl:value-of
select="Numbering"/></xsl:attribute>
    <p>
<b><small>Manufacturer:</small></b>
<br/>
<b> <xsl:value-of select="Manufacturer"/> </b>
<br/>
<b><small>End Product Code:</small> </b>
<br/>
<b> <xsl:value-of select="EndProductCode"/></b>
<br/>
<b><small>End Product Desc.: </small> </b>
<br/>
<b> <xsl:value-of select="EndProductDescription"/></b>
<br/>
<b><small>Color:</small> </b>
<br/>
<b> <xsl:value-of select="Color"/></b>
<br/>
<b><small>Fabric:</small> </b>
<br/>
<b> <xsl:value-of select="Fabric"/></b>
<br/>
<b><small>Quantity:</small> </b>
<br/>
<b> <xsl:value-of select="Quantity"/></b>
<br/>
    </p>
</card>
</xsl:for-each>
</xsl:template>
</xsl:stylesheet>
```

A.2 – *Templates* XSL para a linguagem XHTML

1) Template XSL para a lista de Ordens de Produção do Sistema SC²

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml" doctype-public="-//W3C//DTD XHTML 1.0
Transitional//EN" doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd" indent="yes" />

<xsl:template match="/">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Choose the Production Order</title>
<link rel="stylesheet" type="text/css" href="styles.css" />
</head>
<body>
<p>
<img src="logo_intro2.gif" alt="logo_intro" />
</p>
<p>
<xsl:for-each select="allProductionOrders/productionOrder">
<xsl:sort select="." />
<a>
<xsl:attribute
name="href">/sc2wap/TransformationDomServlet?name=Production_Order&amp;amp;co
de=<xsl:value-of select="/allProductionOrders/@scCode" />|<xsl:value-of
select="@enterpriseCode" />|<xsl:value-of select="@productionOrderCode"
/></xsl:attribute>
<xsl:value-of select="@productionOrderCode" /> (<xsl:value-of
select="@enterpriseName" />)</a>
<br />
</xsl:for-each>
</p>
</body>
</html>
</xsl:template>

</xsl:stylesheet>
```

2) Template XSL para uma Ordem de Produção do Sistema SC²

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="xml"
doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
doctype-system="http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
indent="yes" />

<xsl:template match="/">
  <html xmlns="http://www.w3.org/1999/xhtml">
```

```
<head>
 <title>Production Order Report</title>
 <link rel="stylesheet" type="text/css" href="styles.css" />
            </head>
 <body>
 <xsl:apply-templates />
 <hr/>
 <form action="ChangeDataServlet" method="post">
               <strong>Change Status:</strong>
                <select name="status">
        <option value="CANCELED">CANCELED</option>
        <option value="WAITING">WAITING</option>
        <option value="IN PROGRESS">IN PROGRESS</option>
        <option value="COMPLETED">COMPLETED</option>
        <option value="CREATED">CREATED</option>
<option value="EXECUTED">EXECUTED</option>
         </select>
 <input type="submit" value="Change" />
 <input type="hidden" name="name" value="Production_Order" />
 <input type="hidden" name="code" >
 <xsl:attribute name="value"><xsl:value-of
select="/ProductionOrder/@scCode" />|<xsl:value-of
select="/ProductionOrder/EnterpriseCode" />|<xsl:value-of
select="/ProductionOrder/OrderNumber" /></xsl:attribute>
 </input>
 </form>
</body>
  </html>
</xsl:template>

<xsl:template match="ProductionOrder">

<h2>Product Order Information</h2>
<table>
<tr>
<th>Enterprise Code:</th>
<td>
<xsl:value-of select="EnterpriseCode" />
</td>
</tr>
<tr>
<th>Enterprise Name:</th>
<td>
<xsl:value-of select="EnterpriseName" />
</td>
</tr>
<tr>
<th>Order Number:</th>
<td>
<xsl:value-of select="OrderNumber" />
</td>
</tr>
<tr>
<th>STATUS:</th>
<td>
<xsl:value-of select="Status" />
</td>
</tr>
<tr>
<th>Order Date:</th>
```

```
<td>
<xsl:value-of select="OrderDate" />
</td>
</tr>
<tr>
<th>Delivery Date:</th>
<td>
<xsl:value-of select="DeliveryDate" />
</td>
</tr>
<tr>
<th>Start Date Of Production:</th>
<td>
<xsl:value-of select="StartDateOfProduction" />
</td>
</tr>
<tr>
<th>End Date Of Production:</th>
<td>
<xsl:value-of select="EndDateOfProduction" />
</td>
</tr>

</table>
   <br />
<h2>Production Order Details</h2>
<xsl:apply-templates select="PODetail" />
</xsl:template>

<xsl:template match="PODetail">
<xsl:for-each select="item">
<table>
 <tr>
<th class="title">Item <xsl:value-of select="Numbering" />:</th>
</tr>
<tr>
<th>Product Code:</th>
<td>
<xsl:value-of select="ProductCode" />
</td>
</tr>
<tr>
<th>Requested Quantity:</th>
<td>
<xsl:value-of select="RequestedQuantity" />
</td>
</tr>
<tr>
<th>Produced Quantity:</th>
<td>
<xsl:value-of select="ProducedQuantity" />
</td>
</tr>
<tr>
<th>Misssing Quantity:</th>
<td>
<xsl:value-of select="MisssingQuantity" />
</td>
</tr>
<tr>
```

```
<th>End Product Desc.:</th>
<td>
<xsl:value-of select="RelatedEndProductDescription" />
 </td>
</tr>
</table>
<br />
</xsl:for-each>

</xsl:template>
</xsl:stylesheet>
```

Anexo B – Código de Classes Implementadas

B.1 – Classes do pacote br.ufsc.gsigma.webFramework

1) Classe StartServlet

```
package br.ufsc.gsigma.webFramework;

import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class StartServlet extends HttpServlet {

protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws IOException {

String accept = req.getHeader("accept");
HttpSession session = req.getSession();

try
{
for (Content c : ContentManager.CONTENT_MAP)
{
if (accept.contains(c.getType())) {
String content = c.getName();
ContentManager.storeContent(session, content);
getServletContext().getRequestDispatcher("/"  +  content  +  "/Login."  +
content).forward(req, resp);
break;
}
}
if (ContentManager.getContent(session) == null)
resp.sendRedirect("http://150.162.27.184:8080/sc2");

} catch (ServletException e) {
e.printStackTrace();
throw new RuntimeException(e);
}
}

protected void doPost(HttpServletRequest arg0, HttpServletResponse arg1)
throws ServletException, IOException {
doGet(arg0, arg1);
}
}
```

2) Classe LoginServlet

```
package br.ufsc.gsigma.webFramework;
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class LoginServlet extends HttpServlet {

protected void doGet (HttpServletRequest req, HttpServletResponse resp)
throws IOException {
String login = req.getParameter("Login");
String password = req.getParameter("Password");

HttpSession session = req.getSession();
String content = ContentManager.getContent(session);

if (content == null)
{
resp.setContentType("text/plain");
resp.getWriter().write("Erro! Sessão Expirou");
return;
}

try {
if (loginIsValid(login, password)) {

getServletContext().getRequestDispatcher("/TransformationDomServlet?name
=supplyChains").forward(req, resp);
} else {
getServletContext().getRequestDispatcher("/" + content + "/ErroLogin." +
content).forward(req, resp);
}
} catch (ServletException e) {
e.printStackTrace();
throw new RuntimeException(e);
}
}

private boolean loginIsValid(String login, String password) {
System.out.println(login + " - " + password);
return login.equalsIgnoreCase(password);
}

protected void doPost(HttpServletRequest arg0, HttpServletResponse arg1)
throws ServletException, IOException {
doGet(arg0, arg1);
}
}
```

3) Classe TransformationDomServlet

```
package br.ufsc.gsigma.webFramework;

import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.xml.transform.Result;
import javax.xml.transform.Source;
import javax.xml.transform.Transformer;
```

```
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
import javax.xml.transform.stream.StreamResult;
import javax.xml.transform.stream.StreamSource;

import br.ufsc.gsigma.sc2.webFramework.Sc2XmlManager;

public class TransformationDomServlet extends
javax.servlet.http.HttpServlet implements javax.servlet.Servlet {

private XmlManager xmlManager = new Sc2XmlManager();

protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
doPost(request, response);
}

protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
String content = ContentManager.getContent(request.getSession());

response.setHeader("Cache-Control", "no-cache");
response.setContentType(ContentManager.getContentType(content));

String name = request.getParameter("name");
String code = request.getParameter("code");
String xslt = "/" + content + "/" + name + ".xsl";

File f = new File(getServletContext().getRealPath(xslt));
if (f.exists()) {
try {
TransformerFactory factory = TransformerFactory.newInstance();

InputStream xslStream = getServletContext()
.getResourceAsStream(xslt);

Transformer t = factory.newTransformer(new StreamSource(
xslStream));

Source source = new DOMSource(xmlManager.load(name, code));
Result result = new StreamResult(response.getWriter());
t.transform(source, result);
response.getWriter().flush();
} catch (Exception e) {
e.printStackTrace();
}
} else {
response.setContentType("text/plain");
response.getWriter().write("Arquivo " + xslt + " não existe");
}
}
}
```

4) Classe ChangeDataServlet

```
package br.ufsc.gsigma.webFramework;

import java.io.IOException;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.w3c.dom.Document;

import br.ufsc.gsigma.sc2.webFramework.Sc2XmlManager;
```

```
public class ChangeDataServlet extends javax.servlet.http.HttpServlet
implements javax.servlet.Servlet {

 private XmlManager xmlManager = new Sc2XmlManager();

protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
doPost(request, response);
}

protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException
{
response.setHeader("Cache-Control","no-cache");

String name = request.getParameter("name");
String code = request.getParameter("code");

Document doc = xmlManager.load(name, code);

xmlManager.update(request, doc);

xmlManager.store(doc);

int pos = code.indexOf('|');
if (pos >= 0)
code = code.substring(0, pos);
String file = "/" + ContentManager.getContent(request.getSession()) +
"/Order.jsp?code="+code+"&name="+name;
getServletContext().getRequestDispatcher(file).forward(request,
response);
}

 }
```

5) Interface XmlManager

```
package br.ufsc.gsigma.webFramework;
import javax.servlet.ServletRequest;
import org.w3c.dom.Document;

public interface XmlManager
{

Document load(String name, String code);
void store(Document doc);
void update(ServletRequest request, Document doc);
}
```

6) Classe Content

```
package br.ufsc.gsigma.webFramework;

public class Content
{
private final String name;
private final String type;
public Content(String name, String type)
{
this.name = name;
this.type = type;
}

public String getName() {
return name;
}
```

7) Classe ContentManager

```
package br.ufsc.gsigma.webFramework;
import javax.servlet.http.HttpSession;

public class ContentManager {

public static final Content[] CONTENT_MAP =
    {
    new Content("xhtml", "application/xhtml+xml"),
    new Content("wml", "text/vnd.wap.wml"),
};

private static String lastContent;

public static synchronized void storeContent(HttpSession session, String
content)
{
session.setAttribute("content", content);
lastContent = content;
}
public static synchronized String getContent(HttpSession session)
{
String content = (String) session.getAttribute("content");
if (content == null)
{
System.out.println("Sem *content* na Sessão");
content = lastContent;
}

System.out.println("Content = "+content);

return content;
}
public static String getContentType(String content)
{
for (Content c : CONTENT_MAP) //para cada tipo de conteudo faça:
{
if (c.getName().equals(content))
return c.getType();
}
return null;
}

}
```

B.2 – Classes do pacote br.ufsc.gsigma.sc2.webFramework

1) Interface DocLoader

```
package br.ufsc.gsigma.sc2.webFramework;

import org.w3c.dom.Document;

public interface DocLoader
{

Document load(String code);
}
```

2) Classe SC2XmlManager

```
package br.ufsc.gsigma.sc2.webFramework;

import java.util.HashMap;
import java.util.Map;

import javax.servlet.ServletRequest;

import org.w3c.dom.Document;
import org.w3c.dom.Element;

import br.ufsc.gsigma.sc2.db.ProductionOrderLoader;
import br.ufsc.gsigma.sc2.db.ShippingOrderLoader;
import br.ufsc.gsigma.webFramework.XmlManager;

public class Sc2XmlManager implements XmlManager
{
private static Map<String, DocLoader> docLoaders = null;

public Document load(String name, String code)
{
if (docLoaders == null)
{
docLoaders = new HashMap<String, DocLoader>();
docLoaders.put("supplyChains", new SupplyChainXmlLoader());
docLoaders.put("Customer_Order_List", new CustomerOrderListXmlLoader());
docLoaders.put("Production_Order_List", new
ProductionOrderListXmlLoader());
docLoaders.put("Shipping_Order_List", new ShippingOrderListXmlLoader());
docLoaders.put("Customer_Order", new CustomerOrderXmlLoader());
docLoaders.put("Production_Order", new ProductionOrderXmlLoader());
docLoaders.put("Shipping_Order", new ShippingOrderXmlLoader());
//...
}

return docLoaders.get(name).load(code);
}

public void store(Document doc)
{
Element rootEl = doc.getDocumentElement();
String scCode = rootEl.getAttribute("scCode");
if(rootEl.getNodeName().equals("ShippingOrder"))
{
Element orderNumberEl = (Element)
rootEl.getElementsByTagName("OrderNumber").item(0);
String soCode = orderNumberEl.getFirstChild().getNodeValue();

Element statusEl = (Element)
rootEl.getElementsByTagName("Status").item(0);
String status = statusEl.getFirstChild().getNodeValue();

ShippingOrderLoader.getInstance().updateStatus(scCode, soCode, status);
}
else if(rootEl.getNodeName().equals("ProductionOrder"))
{
Element EnterpriseCodeEl = (Element)
rootEl.getElementsByTagName("EnterpriseCode").item(0);
String enterpriseCode = EnterpriseCodeEl.getFirstChild().getNodeValue();

Element orderNumberEl = (Element)
rootEl.getElementsByTagName("OrderNumber").item(0);
String poCode = orderNumberEl.getFirstChild().getNodeValue();

Element statusEl = (Element)
rootEl.getElementsByTagName("Status").item(0);
```

```
String status = statusEl.getFirstChild().getNodeValue();

ProductionOrderLoader.getInstance().updateStatus(scCode, enterpriseCode,
poCode, status);
}
}

public void update(ServletRequest request, Document doc)
{
String status = request.getParameter("status");

Element rootEl = doc.getDocumentElement();
Element statusEl = (Element)
rootEl.getElementsByTagName("Status").item(0);
statusEl.getFirstChild().setNodeValue(status);
}
}
```

1) Classe ShippingOrderListXmlLoader

```
package br.ufsc.gsigma.sc2.webFramework;

import java.util.List;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;
import org.w3c.dom.Element;

import br.ufsc.gsigma.sc2.beans.adhoc.AllShippingItem;
import br.ufsc.gsigma.sc2.db.AdHocReportsLoader;

public class ShippingOrderListXmlLoader implements DocLoader
{
public Document load(String code)
{
List<AllShippingItem> allShippingOrders =
AdHocReportsLoader.getInstance().loadAllShipping(code);

DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
Document doc = null;
try
{
DocumentBuilder builder = factory.newDocumentBuilder();
doc = builder.newDocument();
Element root = doc.createElement("allShippingOrders");
root.setAttribute("scCode", code);
doc.appendChild(root);
for (AllShippingItem allShippingItem : allShippingOrders)
{
Element scEl = doc.createElement("shippingOrder");
root.appendChild(scEl);
scEl.appendChild(doc.createTextNode(allShippingItem.getShipping()));
}
} catch (ParserConfigurationException e) {
e.printStackTrace();
}

return doc;
}

}
```

1) Interface ShippingOrderXmlLoader

```
package br.ufsc.gsigma.sc2.webFramework;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;

import org.w3c.dom.Document;
import org.w3c.dom.Element;

import br.ufsc.gsigma.sc2.beans.adhoc.ShippingInfo;
import br.ufsc.gsigma.sc2.beans.adhoc.ShippingInfoItem;
import br.ufsc.gsigma.sc2.db.AdHocReportsLoader;

public class ShippingOrderXmlLoader implements DocLoader
{

public Document load(String code)
{
String[] strings = code.split("\\|");
String scCode = strings[0];
String soCode = strings[1];
ShippingInfo shippingInfo =
AdHocReportsLoader.getInstance().loadShippingInfo(scCode, soCode);

DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
Document doc = null;
try
{
DocumentBuilder builder = factory.newDocumentBuilder();
doc = builder.newDocument();
Element root = doc.createElement("ShippingOrder");
root.setAttribute("scCode", scCode);
doc.appendChild(root);
Element element = (Element) root.appendChild(doc.createElement("From"));

element.appendChild(doc.createTextNode(shippingInfo.getManufacturerName(
)));
element = (Element) root.appendChild(doc.createElement("MemberCode"));
element.appendChild(doc.createTextNode(shippingInfo.getManufacturer()));
element = (Element) root.appendChild(doc.createElement("OrderNumber"));
element.appendChild(doc.createTextNode(shippingInfo.getShipping()));
element = (Element)root.appendChild(doc.createElement("ShippingDate"));

element.appendChild(doc.createTextNode(shippingInfo.getShippingDate().to
String()));
element =
(Element)root.appendChild(doc.createElement("PlannedEmbarkDate"));

element.appendChild(doc.createTextNode(shippingInfo.getPlannedEmbark().t
oString()));
element =
(Element)root.appendChild(doc.createElement("RealEmbarkDate"));

element.appendChild(doc.createTextNode(shippingInfo.getRealEmbark().toSt
ring()));
element =
(Element)root.appendChild(doc.createElement("PlannedDeliveryDate"));

element.appendChild(doc.createTextNode(shippingInfo.getPlannedDeliveryDa
te().toString()));
element =
(Element)root.appendChild(doc.createElement("RealDeliveryDate"));

element.appendChild(doc.createTextNode(shippingInfo.getRealDeliveryDate(
).toString()));
```

```
element = (Element)root.appendChild(doc.createElement("To"));
element.appendChild(doc.createTextNode(shippingInfo.getClientName()));
element = (Element)root.appendChild(doc.createElement("MemberCode"));
element.appendChild(doc.createTextNode(shippingInfo.getClient()));
element = (Element)root.appendChild(doc.createElement("Status"));
element.appendChild(doc.createTextNode(shippingInfo.getStatus()));
element = (Element)root.appendChild(doc.createElement("Street"));
element.appendChild(doc.createTextNode(shippingInfo.getStreet()));
element = (Element)root.appendChild(doc.createElement("ZipCode"));
element.appendChild(doc.createTextNode(shippingInfo.getZipCode()));
element = (Element)root.appendChild(doc.createElement("City"));
element.appendChild(doc.createTextNode(shippingInfo.getCity()));
element = (Element)root.appendChild(doc.createElement("Country"));
element.appendChild(doc.createTextNode(shippingInfo.getCountry()));

Element detailElement =
(Element)root.appendChild(doc.createElement("SODetail"));

int i = 1;
for (ShippingInfoItem shippingInfoItem : shippingInfo.getItems())
{
Element itemElement = (Element)
detailElement.appendChild(doc.createElement("item"));
itemElement.setAttribute("id", shippingInfoItem.getShippingDetail());
element = (Element)
itemElement.appendChild(doc.createElement("Numbering"));
element.appendChild(doc.createTextNode(String.valueOf(i++)));
element = (Element)
itemElement.appendChild(doc.createElement("EndProductCode"));
element.appendChild(doc.createTextNode(shippingInfoItem.getProduct()));
element = (Element)
itemElement.appendChild(doc.createElement("EndProductDescription"));

element.appendChild(doc.createTextNode(shippingInfoItem.getNameProduct()
));
element = (Element)
itemElement.appendChild(doc.createElement("Quantity"));

element.appendChild(doc.createTextNode(String.valueOf(shippingInfoItem.g
etQuantity())));
}
} catch (ParserConfigurationException e) {
e.printStackTrace();
}

return doc;
}
}
```

Glossário

A
AES - *Advanced Encryption Standard*
AMPS - *Advanced Mobile Phone Service*
ASCII - *American Standard Code for Information Interchange*
ASP - *Active Server Pages*

B
BREW - *Binary Runtime Enviroment for Wireless*

C
CDMA - *Code Division Multiple Access*
CDPD - *Cellular Digital Packet Data*
cHTML - *compact HTML*
CRM - *Customer Relationship Management*

D
DECT - *Digital Enhanced Cordless Telecommunication*
DES - *Data Encryption Standard*
DISelect - *Content Selection for Device Independence*
DIWG - *Device Independence Working Group*
DTD – *Document Type Definition* ou *Document Type Declaration*

E
EAI - *Enterprise Application Integration*
EJB - *Enterprise JavaBeans*
ERP - *Enterprise Resource Planning*
ETSI - *European Telecommunications Standards Institute*

F
FDMA - *Frequency Division Multiple Access*
FPLMTS - *Future Public Land Mobile Telecommunication System*

G
GSM - *Global System for Mobile Communications*

H
HDML - *Handheld Device Markup Language*
HTML - *Hypertext Markup Language* – Linguagem de Marcação para Hipertexto
HTTP - *Hypertext Transfer Protocol* – Protocolo para Transferência de Hipertexto

I
iDEN - *Integrated Digital Enhaced Network*
IEEE - *Institute of Electrical and Electronic Engineers*
IETF - *Internet Engineering Task Force*
IMT - *International MobileTelecommunications*

IS - *International Standard*
ISM - *Industrial, Scientific and Medical*
ITU - *International Telecommunication Union*

J
JDBC - *Java Database Connectivity*
J2EE - *Java 2 Enterprise Edition*
J2ME - *Java2 Micro Edition*
J2SE - *Java 2 Standard Edition*
JSP - *Java Server Pages*
JTA - *Java Transaction API*

L
LLC - *Logical Link Control*

M
MAC - *Media Access Control* – Controle de Acesso ao Meio
MIME - *Multipurpose Internet Mail Extensions*

N
NMT - *Nordic Mobile Telephone*

O
OSI - *Open Systems Interconnection*

P
PC – *Personal Computer* – Computador Pessoal
PDA - *Personal Digital Assistant*
PDC - *Japanese Personal Digital Cellular*
PHP - *Hypertext Processor*

Q
QoS – *Quality of Service* – Qualidade de Serviço

R
RF- Radiofrequência
RPC - *Remote Procedure Call*
RTT - *Radio Transmission Technology*

S
SC^2 - *Supply Chain Smart Coordination*
SCM - *Supply Chain Management*
SDK - *Software Development Kit*
SGBD - Sistema Gerenciador de Banco de Dados
SIG - *Bluetooth Special Interest Group*
SIM - *Subscriber Identity Module*
SMS - *Short Message Service*

T
TACS - *Total Access Communication System*
TDMA - *Time Division Multiple Access*

U
UMTS - *Universal Mobile Telecommunications System*
URL - *Uniform Resource Locator*
USSD - *Unstructured Supplementary Service Data*

X
XML - *Extensible Markup Language*
XHTML - *Extensible Hipertext Markup Language*
XSL - *Extensible Stylesheet Language*
XSLT – *Extensible Stylesheet Language Transformation*

W
W3C – *World Wide Web Consortium*
WAE - *Wireless Application Environment Layer*
WAG- *Wireless Application Gateway*
WAP - *Wireless Application Protocol*
W-CDMA - *Wideband Code Division Multiple Access*
WDP - *Wireless Datagram Protocol*
WEP - *Wired Equivalent Protection*
Wireless – Sem fio
WLAN - *Wireless Local Área Network*
WML- *Wireless Markup Language*
WPAN - *Wireless Personal Área Network*
WSP - *Wireless Session Protocol Layer*
WTLS - *Wireless Transport Layer Security*
WTP- *Wireless Transaction Protocol*
WWAN - *Wireless Wide Area Network*

Referências Bibliográficas

AIR2WEB; 2003. *Mobile Internet Platform 5.0, Data Sheet*. www.air2web.com., acessado em 20 de junho de 2003.

BICKMORE, T.; SCHILIT, B. N.; 1997. Digestor: Device-independent Access to the World Wide Web. In: *Computer Networks and ISDN Systems*, vol. 29, n. 8-13, p. 1075-1082.

BRADESCO; 2005. http://www.bradesco.com.br/br/universob/?paramPag=UnivBrad, acessado em 20 de dezembro de 2005.

BRANS, P.; 2002. *Mobilize Your Enterprise: Achieving Competitive advantage Through Wireless Technology*. Prentice Hall. ISBN 0-13-009116-2.

BREW; 2004a. http://brew.qualcomm.com/brew/en/img/about/pdf/bds.pdf, acessado em 28 março de 2005.

BREW; 2004b. http://www.wave-report.com/tutorials/brew.htm, acessado em 28 março 2005.

CASTALDELLI, M.; 2003. Aplicações Atuais e Futuras para Internet Móvel. www.teleco.com.br/tutoriais/tutorialcmovel/default.asp, acessado em 30 de junho de 2005.

DABKOWSKI, A.; JANKOWSKA, A. M.; KURBEL, K.; 2003. Designing Global Applications for Wireless Devices with Java e XML. In: INTERNATIONALIZATION AND UNICODE CONFERENCE (IUC23) (23: 24-26 Mar. 2003: Prague, Czech Republic). *Proceedings.* Prague, Czech Republic. p. 1-15.

DEITEL, H.M.; DEITEL, P, J.; tradução: FUMANKIEWICZ, E.; 2001. *JAVA Como Programar*. 3. ed. Porto Alegre: Bookman. ISBN 85-7307-727-1

DUBENDORF, V. A.; 2003. *Wireless Data Technologies*. Wilwey. ISBN 0470849495

ECLIPSE; 2005. www.eclipse.org, acessado em 11 de outubro de 2005.

ENTERPRISE SOFTWARE; 2005. http://www.etp.com.br/wwwpt/details.asp, acessado em 19 de julho de 2005.

FORMAN, G. H.; ZAHORJAN, J.; 1994. The Challenges of Mobile Computing. *IEEE Computer*, USA, v.27, n. 4, (Apr.) p.38-47.

GEIER, J.; 2001. *Wireless LAN`s.* 2nd edition. USA: Sams Publishing. ISBN 0672320584.

GÖBEL, S.; BUCHHOLZ, S.; ZIEGERT, T. et al; 2001. Device Independent Representation of Web-based Dialogs and Contents. In: IEEE YOUTH FORUM IN COMPUTER SCIENCE AND ENGINEERING (YUFORIC'01) (1: 29 – 30 Nov. 2001: Valencia, Spain). *Proceedings*. Valencia, Spain.

GSIGMA; 2006. http://www.gsigma.ufsc.br/index.html, acessado em 20 de abril de 2006.

GUIMARÃES, D. A.; 2001. Sistemas de Comunicação Móvel de Terceira Geração. *Telecomunicações*, Minas Gerais, v.4, n.1 (Maio), p. 1-23. ISBN 1516-2338 http://www.inatel.br/revista/volume-04-n1

HARRTSEN, J.; ALLEN, W.; INOUYE, J. et al.; 1998. Bluetooth: Vision, Goals, and Arquiteture. *ACM Mobile Computing and Communications Review*, USA, v. 4, n.2, p.38-45.

HEIJDEN, M. V.; TAYLOR, M.; 2000. *Understanding WAP: Wireless Applications, Devices, and Services*. Artech House. ISBN 1580530931

IBM; 2005. http://www-306.ibm.com/software/pervasive/transcoding_publisher/, acessado em 28 de agosto de 2005.

IEEE, 2006. http://www.computer.org/portal/site/pervasive/, acessado em 10 de abril de 2006.

IEEE 802.11. (http://grouper.ieee.org/groups/802/11/index.html, acessado em 01 de novembro de 2005.

IETF; 2005. www.ietf.org, acessado em 05 de dezembro de 2005.

I-MODE; 2005. http://e-docs.bea.com/wls/docs81/wireless/imode.html, acessado em 20 de julho de 2005.

JAVA TECHNOLOGY; 2005.java.sun.com, acessado em 12 de novembro de 2005.

JAVA EE; 2006. http://java.sun.com/javaee/, acessado em 15 de abril de 2006.

KDDI; 2005. http://www.au.kddi.com/english/ezweb/index.html, acessado em 15 de dezembro de 2005.

KIMURA, T.; KANDA, Y; 2003. Development of a Remote Monitoring System Using a Portable Phone for Manufacturing Support Systems. In: INTERNATIONAL CONFERENCE ON ADVANCES IN PRODUCTION MANAGEMENT SYSTEMS (IFIP TC5/WG5.7) (8: 8-13 Sep. 2002: Eindhoven, The Netherlands). *Proceedings.* Eindhoven, The Netherlands. p. 243-254.

KWOK, T.; NGUYEN, T.; LAM, L. et al.; 2004. An Efficient and Systematic Method to Generate XSLT Stylesheets for Different Wireless Pervasive Devices. In: INTERNATIONAL WORLD WIDE WEB CONFERENCE ON ALTERNATE TRACK PAPERS & POSTERS (13: 19 – 21 May 2004: New York, USA). *Proceedings.* New York, USA. p. 218-219.

MACRO; 2005. *Mobility Assistance for Customer Relations Based Organisations*. http://vega.sunderland.ac.uk/macro/, acessado em 14 de junho de 2005.

MELLO, B.; REIS, S.; 2005. Mobilidade e Sobrevivência no E-business. http://www.b2bmagazine.com.br/ler_materia.aspx?numero=12903, acessado em 20 de julho de 2005.

MOBILE COMMERCE; 2005. http://www.vg-u.de/euv-new-site/descr_de_mc.asp, acessado em 30 de junho de 2005.

NAKAMURA, E.; FIGUEIREDO, C. M.S.; 2003. Computação Móvel: Novas Oportunidades e Novos Desafios. 16. *T&C Amazônia*, Ano 1; n.2; junho de 2003.

NOKIA; 2005. http://direct.www.forum.nokia.com/main/1,6566,034-13,00.html, acessado em 08 de agosto de 2005.

Ntt DoCoMo; 2005a. http://www.nttdocomo.com/companyinfo/subscriber.html, acessado em 10 de dezembro de 2005.

NYLANDER, S; BYLUND, M.; 2002. Providing Device Independence to Mobile Services. SICS,*Technical Report* - T2002-02A, *Kista, SWEDEN.*

OMA; 2005.http://www.openmobilealliance.org/, acessado em 03 de outubro de 2005.

OPENWAVE; 2005. http://developer.openwave.com/dvl/tools_and_sdk/phone_simulator/, acessado em 05 de setembro de 2005.

PIGNEUR, Y; CAMPONOVO, G.; 2003. Business Model Analysis Applied to Mobile Business. In: INTERNATIONAL CONFERENCE ON ENTERPRISE INFORMATION SYSTEMS (ICEIS 2003). (5: 22-26 Apr. 2003: Angers, France). *Proceedings.* Angers, France. p.173-183.

PEREIRA-KLEN, A.A.; RABELO, R. J., 2003. Deliverable D11.B - *Dynamic Supply Network Coordination Specification*. Universidade Federal de Santa Catarina.

RABELO, R. J.; PEREIRA-KLEN, A.; KLEN, E. R.; 2002. A Multi-Agent System for Smart Coordination of Dynamic Supply Chains. In: IFIP WORKING CONFERENCE ON INFRASTRUCTURES FOR VIRTUAL ENTERPRISES (PRO-VE'02) (3: 1-3 May 2002: Sesimbra, Portugal). *Proceedings*. Sesimbra, Portugal. p. 379-386.

ROBINSON, M.; KALAKOTA, R.; 2001. M-business: The Race To Mobiliy. *Business Integration Journa*l, Dallas, Dec. http://www.bijonline.com/Article.asp?ArticleID=471

ROSENBERG, A.; KEMP, S.; 2002. *CDMA Capacity and Quality Optimization*. McGraw-Hill. ISBN 0071399194.

SIVALINGAM, K.M.; JONES, C.E.; AGRAWAL, P. et al.; 2001. A Survey of Energy Efficient Network Protocols for Wireless Networks. *Wireless Networks*, Hingham, v.7, n.4 (Aug.), p. 343-358.

SMS; 2005. http://www.iec.org/online/tutorials/wire_sms/index.html, acessado em 22 de março de 2005.

SOARES, L. F.; LEMOS, G.; COLCHER, S.; 1997. *Redes de Computadores:das LANs, MANs e WANs às Redes ATM*. 2 ed. Editora Campus ISBN 85-700-1998-X.

SONERA MEDIALAB; 2002. *Mobile Search Engines White Paper*. http://www.medialab.sonera.fi/workspace/MobileSearchEnginesWhitePape.pdf, acessado em 23 de setembro de 2005.

STANOEVSKA-SLABEVA, K; 2003. Towards a Reference Model for M-Commerce Applications. In: EUROPEAN CONFERENCE ON INFORMATION SYSTEMS (ECIS 2003) (11: 16-21 June 2003: Naples, Italy). *Proceedings.* Naples, Italy. p. 1-13.

TANEMBAUM A.S.; 2003. *Computer Networks.* 4 edition. USA: Prentice-Hall, ISBN 0-13-066102-3.

TEIXEIRA JUNIOR; 2005. Ele Mudou até a Vida. *Revista EXAME.* Abril; Ed. 844; Ano 39; n.11; 8 de junho de 2005.

TELECA; 2005. http://www.teleca.se/PSUser/mediacache/External_media/9791_1.pdf, acessado em 28 de junho de 2005.

TOPLEY, K.; 2002. *J2ME in a Nutshell*. O'Reilly. ISBN 059600253X

TRAMONTIN, R. J.; 2004. *Configuração e Integração de Dados em Plataformas para Empresas Virtuais.* (Mestrado em Engenharia Elétrica) - Centro Tecnológico, Universidade Federal de Santa Catarina.

TT; 2005. http://emulator.tagtag.com/wapemulator.cgi, acessado em 05 de setembro de 2005.

TWIC; 2005. http://www.twic.com.br/handbusiness/, acessado em 30 de junho de 2005.

UNISANTA; 2005. http://www.online.unisanta.br/2005/09-17/index.htm, acessado em 23 de setembro de 2005.

VARSHNEY, U.; 2003. The Status and Future of 802.11-based WLANs. *IEEE Computer*, USA, v. 36, n. 6, (June) p. 102-105.

VODAFONE; 2005. http://www.vodafone.jp/en/live/index.html, acessado em 21 de dezembro de 2005.

WAPEDUC; 2005. http://psteger.free.fr/newsite/index.html, acessado em 25 de setembro de 2005.

WAP PROFF; 2005. http://www.wap-proof.com/, acessado em 05 de setembro de 2005.

W3C; 2001. Extensible Stylesheet Language (XSL)Version 1.0. W3C Recommendation 15 October 2001. http://www.w3.org/TR/xsl/

W3C; 2002. Extensible Hipertext Markup Language (XHTML)Version 1.0. W3C Recommendation 01 August 2002. http://www.w3.org/TR/xhtml1/

W3C; 2004. Content Selection for Device Independence (DISelect) Version 1.0. W3C Recommendation 11 june 2004. http://www.w3.org/TR/2004/WD-cselection-20040611/

WEBB, W.; 1999. *The Complete Wireless Communications Professional*. Artech House. ISBN 0890063389.

WILMA; 2001. *Wireless Internet and Location Management Architecture*. http://www.wilmaproject.org/aims.html, acessado em 16 de junho de 2005.